

Research Article

Unified Spatial Intersection Algorithms Based on Conformal Geometric Algebra

Feng Zhang,^{1,2} Xiaomin Jiang,² Xiaoyi Zhang,² Yingzhi Wang,³
Zhenhong Du,^{1,2} and Renyi Liu^{1,2}

¹Zhejiang Provincial Key Laboratory of Geographic Information Science, Zhejiang University, 148 Tianmushan Road, Hangzhou 310028, China

²School of Earth Sciences, Zhejiang University, 38 Zheda Road, Hangzhou 310027, China

³Department of Public Order, Zhejiang Police College, 555 Binwen Road, Hangzhou 310053, China

Correspondence should be addressed to Zhenhong Du; duzhenhong@zju.edu.cn

Received 11 May 2016; Revised 1 September 2016; Accepted 29 September 2016

Academic Editor: Eckhard Hitzer

Copyright © 2016 Feng Zhang et al. This is an open access article distributed under the Creative Commons Attribution License, which permits unrestricted use, distribution, and reproduction in any medium, provided the original work is properly cited.

Conformal Geometric Algebra has been introduced into geographic information science as a mathematical theory because of its advantages in terms of uniform multidimensional representation and computation. The traditional intersection computation between two geometric objects of different types is not unified. In this study, we propose algorithms based on Conformal Geometric Algebra to determine the spatial relationships between geographic objects in a unified manner. The unified representation and intersection computation can be realized for geometric objects of different dimensions. Different basic judgment rules are provided for different simple geometries. The algorithms are designed and implemented using MapReduce to improve the efficiency of the algorithms. From the results of several experiments we provide, the correctness and effectiveness of the algorithms can be verified.

1. Introduction

Geographic information is used widely to support digital replicas of the real world. For decades, spatial objects and their relationships have been represented and calculated based on Euclidean geometry. Due to the variable requirements for the expression and computation of multidimensional objects, researchers have aimed to improve or find new ways to satisfy needs, including the mathematical foundations employed [1]. Thus, Liu et al. proposed a general-field model to unify continuous field and discrete object conceptualizations [2]. Li and Qian presented a combined dynamic binary generalization tree method to represent two-dimensional geographic objects and for managing spatial data based on volunteered geographic information [3]. Yang et al. proposed a multiscale expression-oriented subdivisional encoding method to meet the demands of geographic object representation at different application levels, with different details, and arbitrary scales [4].

The feature of multidimensional-unified representation makes Geometric Algebra (GA) become a hot research in

geography. Yuan et al. proposed a multidimensional-unified data model based on Clifford algebra and they developed a prototype software system called Clifford algebra based on unified spatial-temporal analysis (CAUSTA) for investigating and modeling the distribution characteristics and dynamic processes of complex geographical phenomena [5]. GA has been employed widely as a new mathematical tool for multidimensional-unified representation and computation [6–15]. Luo et al. developed a new data structure to support unified organization and computation of geometrical primitives, which can reduce the complexity of data architectures and improve the processing ability of computer graphic software, but the extra two dimensions in Conformal Geometric Algebra could bring in low efficiency [16]. Yu et al. implemented multidimensional representation for 3D vector data, and calculated intersection relations between Delaunay-Triangulated Irregular Networks (DTINs) with meet operator. They conducted parallel computation by GPU to improve computing efficiency [17]. In this paper, we use meet operator to judge intersection relation between objects of different types and MapReduce to implement parallel computing.

Many scholars do some researches on the triangle-to-triangle intersection test. In [18], Möller proposed a signed distance method to exclude some nonintersection situations and detect intersection if two segments between one triangle and the plane to which the other triangle belongs overlap. In [19], based on signed distance and the intersection relationship between the two intersection line segments, Held gave an algorithm for different kinds of geometric objects. From the algebraic perspective, Tropp et al. made full use of the process values during the problem resolving and the linear relation in matrix operation to accelerate the problem to be resolved [20]. Guigue and Devillers employed the signed determinant and the intersection line relationship to determine the intersection relationships of the two triangles [21]. In [22], based on Conformal Geometric Algebra, Zong et al. discussed the intersection computation between two triangles and verified the algorithm by using Antarctic ice sheet simulation data. Luo et al. resolved the intersection relationship of two triangles based on *MVTree* structure, but some optimization methods may be used to increase some efficiency in the future [16]. In [23], Shen et al. implemented a fast detection algorithm for 3D convex polygons based on clamping-edge pair. Wei presented an algorithm which absorbed the idea from Tropp reducing the intersection test to the segment and triangle in the same planar and used the idea of classified discussions by comparing directions of coplanar vectors' cross products [24]. Sugihara developed an intersection algorithm based on Delaunay triangulation [25]. Most of them show intersect in the intersection algorithm between two triangles and test the intersection relationship between two polygons by decomposing them into triangles.

However, the above studies mainly aimed at the intersection relationship judgment between two triangles. The algorithms dealing with geometric objects of different types are rare and not unified.

The computation of spatial relationships is the most basic operation in spatial analysis, including topological, order, and metric relationships. The intersection algorithm is a basic operator for topological computation. The parallel line segment intersection strategy based on uniform grids was developed for use in a shared architecture, which can effectively utilize computational resources in a parallel manner [26]. Wang proposed a parallel intersection algorithm for vector polygon overlay [27]. The parallel computation strategy can work well when coping with large volumes of data.

In the present study, we attempt to handle geometric objects of different types in a unified fashion based on Conformal GA (CGA) and to expand the algorithm on the MapReduce platform to process massive amounts of spatiotemporal data.

The main contributions of the present study are as follows.

- (1) We propose unified spatial algorithms and specific rules for intersection relationships judgment for geometric objects of different types.
- (2) We design several experiments to verify the correctness of the unified intersection algorithms. And the algorithms are implemented on the MapReduce

platform to overcome the restriction of using hardware on standalone mode.

2. Methodology

2.1. CGA. GA is an algebraic language that solves geometric problems in algebraic form. The geometric product unifies the dimension, geometry, vector, and scalar operations [28–30].

Suppose that we have two arbitrary multivectors, A , B ; then, the geometric product between A and B is

$$AB = \left(\sum_{s=1}^n \langle A \rangle_s \right) B = \sum_{s=1}^n (\langle A \rangle_s B), \quad (1)$$

where A denotes a linear combination of its homogeneous grades parts, n denotes the dimension of A , $\langle A \rangle_s$ denotes multivector part of grade s . The product can be decomposed into basic geometric products of grade 1 vectors [31, 32].

CGA is the most common GA space. Based on the traditional Euclidean space, two extra base vectors, e_o and e_{∞} , are employed to represent basic geometric objects. The hierarchical geometric structure corresponds to the hierarchical Grassmann structure for the outer product in CGA. The inner product in CGA can describe basic metric information such as the distance and angle. We construct a multidimensional-unified data model based on *blade* and *multivector* and intersection algorithms using the basic operations in CGA.

2.2. Multidimensional-Unified Representation. A *blade* is the outer product of a series of linearly independent vectors. A *blade* with grade k can be represented by k -*blade*. When grade is 0, 1, 2, 3, 4, and 5, the corresponding *blade* is scalar, vector, bivector, trivector, quadvector, and pseudoscalar, respectively [29].

A *multivector* is the linear combination of geometric objects with different dimensions and it is the basic mathematical structure for multidimensional objects. The *multivector* can be utilized to represent complex geometry objects and it contributes to making objects with different dimensions representation-unified.

The grade extracting operation $\langle \rangle_i$ is used to resolve the part with grade i in the *multivector*. For example, in the 5 dimensions conformal space, for example, for $A = a_0 + a_1 e_1 + \dots + a_{12} e_{12} + \dots + a_{123} e_{123} + \dots + a_{1234} e_{1234} + \dots + a_{12345} e_{12345}$, the resolution results are as follows:

$$\begin{aligned} \langle A \rangle_0 &= a_0 && \text{Scalar} \\ \langle A \rangle_1 &= a_1 e_1 + \dots && \text{Vector} \\ \langle A \rangle_2 &= a_{12} e_{12} + \dots && \text{Bivector} \\ \langle A \rangle_3 &= a_{123} e_{123} + \dots && \text{Trivector} \\ \langle A \rangle_4 &= a_{1234} e_{1234} + \dots && \text{Quadvector} \\ \langle A \rangle_5 &= a_{12345} e_{12345} && \text{Pseudoscalar.} \end{aligned} \quad (2)$$

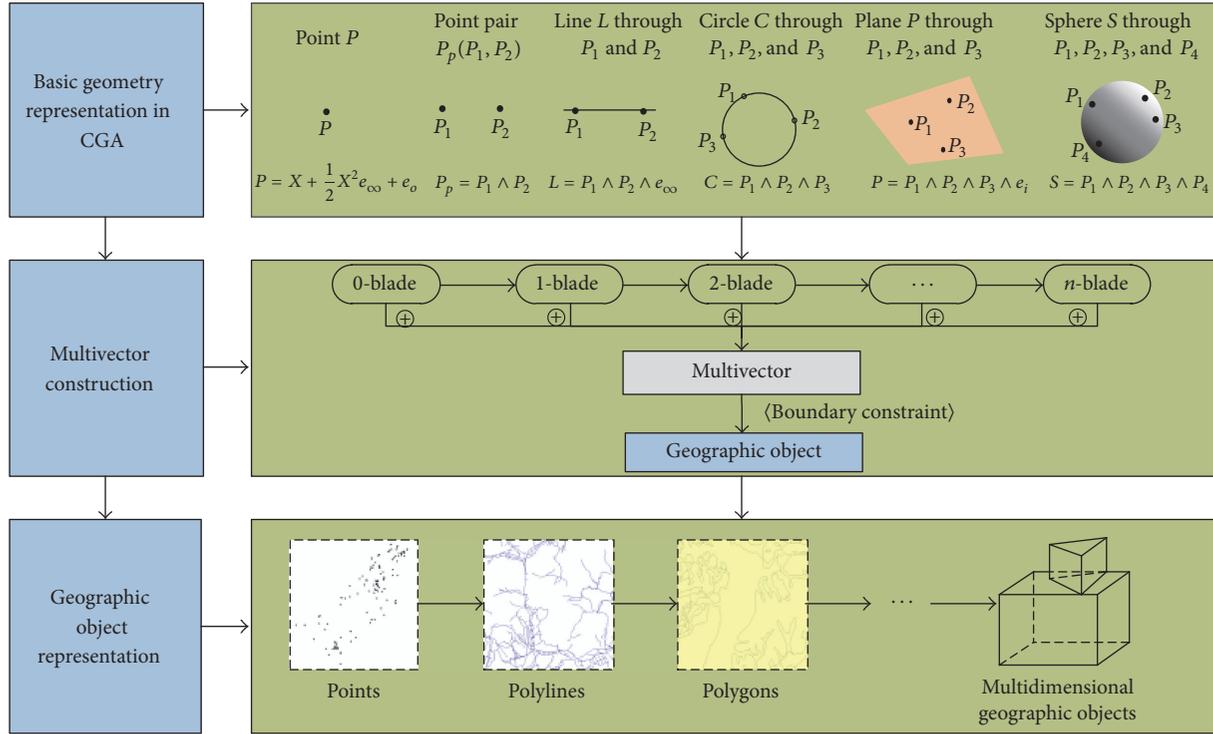


FIGURE 1: Multidimensional-unified representation data model process based on the CGA.

For instance, given two points $P_1(1, 0, 0)$ and $P_2(0, 1, 0)$, the representations in conformal space are $P_1 = e_1 + (1/2)e_\infty + e_0$ and $P_2 = e_2 + (1/2)e_\infty + e_0$. The line constructed by the two points is $L_{P_1P_2} = P_1 \wedge P_2 \wedge e_\infty$. Using the outer product, the operation is $L_{P_1P_2} = e_1 \wedge e_2 \wedge e_\infty + e_1 \wedge e_0 \wedge e_\infty + e_0 \wedge e_2 \wedge e_\infty$, which is part of A and corresponding to $\langle A \rangle_3$ in the grade resolution. For other basic objects, for example, the point can be resolved using $\langle A \rangle_1$, the point pair $\langle A \rangle_2$, the line and circle $\langle A \rangle_3$, and the plane and sphere $\langle A \rangle_4$. Thus, the *multivector* is capable of representing geometry with different dimensions. In this study, a geographic object is represented by a *multivector* with boundary constraints. The data modeling process is illustrated in Figure 1.

We can represent an arbitrary geometric object in the following CGA format:

$$[\text{ObjectType}, \text{ObjectID}, \text{ObjectFormat}, \text{Attr1}, \text{Attr2}, \dots, \text{Timestamp}] \langle \text{constraint} \rangle, \quad (3)$$

where *ObjectType* represents the type of the object, *ObjectID* represents the unique identifier, *ObjectFormat* represents the multivector representation, *Attr1*, *Attr2*... represent attribute information for the object, *Timestamp* represents the time information, and $\langle \text{constraint} \rangle$ is the boundary constraint. When the *ObjectType* is a point, *ObjectID* is the point ID. When the *ObjectType* is a line segment, $\langle \text{constraint} \rangle$ denotes the ID of the endpoints of the line segment. When the *ObjectType* is a polygon, $\langle \text{constraint} \rangle$ denotes the ID of the boundary lines of the polygon.

2.3. Intersection Algorithm

2.3.1. Meet Operator. The meet operator can be utilized to determine the intersection relationship between two objects of different dimensions in a unified fashion. The objects that the meet operator can deal with directly are basic geometric objects with no boundaries.

Given two blades, A and B , the meet operator satisfies

$$\forall x : \begin{cases} x \wedge A = 0 \\ x \wedge B = 0. \end{cases} \quad (4)$$

The concrete computing formula is as follows:

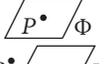
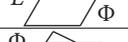
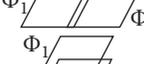
$$M = A \cap B = (B \cdot I_{AB}) \cdot A = B^* \cdot A, \quad (5)$$

where I_{AB} denotes the *pseudoscalar* forming the minimal subspace which contains A and B . B^* is the duality operation of B . In [16], Luo et al. employed meet operator to judge intersection relationships.

Under certain circumstances, the objects are constrained with their boundaries. The specific intersection relationship judgment rules are depicted in Section 2.3.2.

2.3.2. Specific Judgment Rules. Cameron and Lasenby stated that CGA can provide benefits for understanding multivectors and improving algorithms for computer graphics and vision. They discuss intersection ordering for from/back-face to investigate possible advantages to show different spheres and triangular facets in a scene. Here, we try to construct a comprehensive inclusion relationship between basic geometric objects [33]. Different intersection algorithms are

TABLE 1: The meet operation results between basic objects.

$Re\ s = P_1 \cap P_2$	If $Re\ s = \text{point}$, P_1, P_2 intersect	$P_1 \bullet P_2$
	If $Re\ s = \text{MGAO}$, P_1, P_2 do not intersect	$P_1 \bullet \bullet P_2$
$Re\ s = P \cap L$	If $Re\ s = \text{point}$, P, L do not intersect	$L \text{---} \bullet P \text{---}$
	If $Re\ s = \text{MGAO}$, P, L do not intersect	$L \text{---} \bullet P$
$Re\ s = P \cap \Phi$	If $Re\ s = \text{point}$, P, Φ intersect	
	If $Re\ s = \text{MGAO}$, P, Φ do not intersect	
$Re\ s = L_1 \cap L_2$	If $Re\ s = \text{point}$, L_1, L_2 intersect	
	If $Re\ s = \text{line}$, L_1, L_2 intersect	
	If $Re\ s = \text{MGAO}$, L_1, L_2 do not intersect	
$Re\ s = L \cap \Phi$	If $Re\ s = \text{point}$, L, Φ intersect	
	If $Re\ s = \text{line}$, L, Φ intersect	
	If $Re\ s = \text{MGAO}$, L, Φ do not intersect	
$Re\ s = \Phi_1 \cap \Phi_2$	If $Re\ s = \text{point}$, Φ_1, Φ_2 intersect	
	If $Re\ s = \text{plane}$, Φ_1, Φ_2 intersect	
	If $Re\ s = \text{MGAO}$, Φ_1, Φ_2 do not intersect	

employed for different types of geometric objects. For the intersection judgment, the outer product, inner product, and geometric product are employed.

First, the line segment is the basic construction unit for a polygon. The complex geometric objects are decomposed into several simple geometric objects to perform the intersection operation. The meet operation results between basic objects are as seen in Table 1. Here, we define MGAO (Meaningless Geometric Algebra Object), which refers to the objects that are not meaningful in GIS. If the computation result is recorded as MGAO that means that the basic GA objects are not intersected.

Case 1 (the intersection relationship between two points P_1 and P_2). Record meet operation result $Re\ s$ between the two points P_1 and P_2 . If $Re\ s$ is a point, P_1 intersects with P_2 , and If $Re\ s$ is MGAO, P_1 does not intersect with P_2 .

Case 2 (the intersection relationship between point P and line segment L_{AB}). Line segment L_{AB} is a line with a boundary constraint. Straight line L_{SL} is the line going through line segment L_{AB} . Record, meet operation result $Re\ s$ between point P and straight line L_{SL} .

If $Re\ s$ is MGAO, P and L_{AB} do not intersect. If $Re\ s$ is a point, then, for line L_{SL} containing A and B , the outer product representation is $L_{SL} = A \wedge B \wedge e_{\infty}$. We need to determine the intersection relationship between point P and line segment L_{AB} . According to the inner product, outer

product, and geometric product in GA, we can construct two representations, as follows:

$$\begin{aligned} t_1 &= (A \wedge P \wedge e_{\infty}) L_{SL} \\ t_2 &= (P \wedge B \wedge e_{\infty}) L_{SL}. \end{aligned} \quad (6)$$

If $t_1 = 0 \parallel t_2 = 0$, then point P matches one of the endpoints of line segment L_{AB} , which means that P intersects with L_{AB} .

If $t_1 < 0 \parallel t_2 < 0$, then point P is not within line segment L_{AB} , which means that P does not intersect with L_{AB} .

Case 3 (the intersection relationship between point P and polygon M). Record the meet operator result $Re\ s$ between point P and polygon M . If $Re\ s$ is MGAO, P and M do not intersect. If $Re\ s$ is a point, then we construct the representation: $t = (P \wedge L) \cdot \Phi$, where P is the point that needs to be assessed, L is the boundary of the polygon, Φ is the plane to which M belongs, $\Phi = C \wedge e_{\infty}$, C is the circle that surrounds polygon M , and e_{∞} is infinity.

We obtain the boundary of polygon M sequentially and if equation $t < 0 \parallel t = 0$ meets each boundary, then point P is within or on the boundary of polygon M (that is P intersects with M); otherwise, point P is outside polygon M (i.e., P does not intersect with M).

Case 4 (the intersection relationship between two line segments $L_{A_1B_1}$ and $L_{A_2B_2}$). For line L_{SL_1} containing A_1 and B_1 ,

$L_{SL_1} = A_1 \wedge B_1 \wedge e_{\infty}$, and, for the line L_{SL_2} containing A_2 and B_2 , $L_{SL_2} = A_2 \wedge B_2 \wedge e_{\infty}$. By transferring the entities from Euclidean space into conformal space, the following equation can be obtained:

$$\begin{aligned} L = & L_1 e_{012} + L_2 e_{013} + L_3 e_{01\infty} + L_4 e_{023} + L_5 e_{02\infty} \\ & + L_6 e_{03\infty} + L_7 e_{123} + L_8 e_{12\infty} + L_9 e_{13\infty} \\ & + L_{10} e_{23\infty}. \end{aligned} \quad (7)$$

By using the meet operation, we have the following:

$$\text{Meet}(L_{SL_1} L_{SL_2}) = L_{SL_1} \cap L_{SL_2} = (L_{SL_2} \cdot I) \cdot L_{SL_1}. \quad (8)$$

This operation yields the multivector, MGAO, or a point object.

Step 1. The meet operation is used to compute the intersection result between two lines. If the result is MGAO, then the two lines are parallel; otherwise, go to Step 2.

Step 2. Assess the intersection relationship between the intersection point and the two line segments, according to the detailed rules described in Case 2.

Step 3. If the intersection point is within or matches with one endpoint of the two line segments, then the two line segments intersect.

Case 5 (the intersection relationship between line segment L and polygon M). Construct straight line L_{SL} going through the line segment and plane Φ covers the polygon. Record the meet operator result $\text{Re } s$ between L_{SL} and Φ .

If $\text{Re } s$ is MGAO, L_{SL} and Φ do not intersect and the line segment L and the polygon M do not intersect either. If $\text{Re } s$ is a line, L and M intersect. If $\text{Re } s$ is a point, named Q , then, according to Case 2, we obtain the intersection relationship IR_1 between point Q and line segment L . According to Case 3, intersection relationship IR_2 between point Q and polygon M is achieved. If IR_1 and IR_2 meet the conditions simultaneously, L and M intersect; otherwise, L and M do not intersect.

Case 6 (the intersection relationship between two polygons M_1 and M_2). We construct two planes using three points that are not collinear and infinity; that is, $\Phi_1 = A_1 \wedge B_1 \wedge C_1 \wedge e_{\infty}$ and $\Phi_2 = A_2 \wedge B_2 \wedge C_2 \wedge e_{\infty}$. Record the meet operator result $\text{Re } s$ between two planes Φ_1 and Φ_2 .

If $\text{Re } s$ is a plane, it means that the two polygons are coplanar. The intersection relationship judgment is a coplanar problem. The detailed judgment rule is as follows.

For the two polygons, the representations are $\Sigma_1 = A_1 \wedge B_1 \wedge C_1 \wedge e_{\infty} \langle A_1, B_1, C_1, \dots \rangle$ and $\Sigma_2 = A_2 \wedge B_2 \wedge C_2 \wedge e_{\infty} \langle A_2, B_2, C_2, \dots \rangle$, respectively. The IDs in angle bracket are boundary constraints. First, we assess the intersection relationship between the vertices of one polygon and the other polygon (see Case 3). If the intersection relationship is not satisfied, we assess the intersection relationship between

the boundary line segments of the two polygons (see Case 4). Thus, we can summarize the assessment steps as follows.

Step 1. Assess the intersection relationship between the vertices of these two polygons. If there is one vertex within the other polygon, the two polygons intersect; otherwise, go to Step 2.

Step 2. Decompose the polygon into a boundary line set.

Step 3. According to Step 1, we have taken the situation where one object is inside the other one into consideration. Then, for each line segment in one polygon, we obtain its intersection relationship with each line segment in the other polygon. If there is an intersection relationship between the two line segments, then the two polygons intersect; otherwise, they do not intersect with each other.

If $\text{Re } s$ is a straight line, named L_{SL} , we have to determine the intersection relationship between L_{SL} and Φ_1, Φ_2 , respectively.

For each line segment in polygon M_1 , we obtain its intersection relationship with straight line L_{SL} (see Case 5). Also, we obtain the intersection relationship between each line segment of polygon M_2 and straight line L_{SL} . If both of these two situations meet, the two polygons intersect; otherwise, they do not intersect with each other.

The above assessments are specific judgment rules for testing intersection relationships. To cope with massive amounts of spatial data, we must implement the intersection algorithms in parallel. However, one computer always lacks sufficient computational resources for this purpose in terms of its CPU and memory, especially when the target data reach a specific volume. Thus, it is obviously difficult for a standard PC to run a massive data computation program and it is also time-consuming, along with being unsuitable in certain conditions. In many studies, parallel computing has been performed using massively parallel processors, symmetric multiprocessors, OpenMP, or clusters. The computational power can be enhanced and the time required can be decreased, but the hardware on one personal computer is still limited. Thus, we used the open source parallel implementation of MapReduce to achieve our aim.

3. Case Study

The intersection algorithms are implemented on open source codes GA sandbox 1.0.7. The basic operations, such as inner product, outer product, and geometric product, and the basic operators, like reverse, reflection, meet, join, and other operators, are implemented. So, it is convenient for us to accomplish our work.

The data we used in our experiments are land use data, which are common vector data, such as shp, mdb, and gdb. We take land use data to test parallel intersection algorithms, the numbers of the 5 features sets are as follows: 22183, 174881, 695343, 1386987, and 5547948. In addition, the data we referred to in parallel experiments are coplanar.

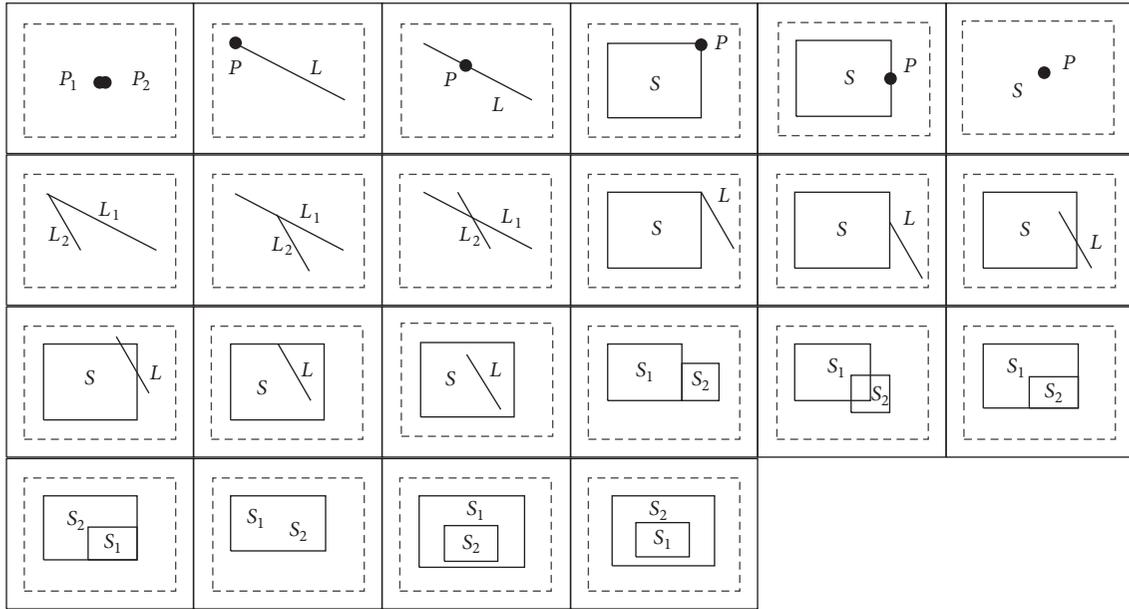


FIGURE 2: Intersection relationships between geometries of different types.

3.1. The Correctness Verified Experiments. Point, polyline, and polygon are three basic types of geometries with which we often deal in spatial analysis. We are often confronted with the intersection computing among them. As described in Section 2.3, we can implement the intersection computing in a unified manner. In Figure 2, intersection relationships between geometries of different types that can be distinguished are depicted.

In Figure 2, using intersection algorithms, we can judge whether two points intersect. For one point and one line segment, we recognize the intersection relation meets when the point locates the edge or interior of the line segment. For one point and one polygon, as long as the point is situated in the edge or interior of the polygon, they intersect. For two line segments, we judge the location of intersecting point; if the intersecting point lies in the edge or interior of both line segments, they intersect. For one line segment and one polygon, we have to judge whether the intersecting point lies in the edge or interior of the line segment and, one special situation, that is, whether one edge of the line segment is inside the polygon. For two polygons, the 7 intersection situations can be distinguished according to the intersection algorithms.

To verify the correctness of the algorithms, we design a series of experiments. In Figure 3, the unified intersection algorithms are carried out. The features of different types intersecting with the region in red color are selected. Experiments show that the unified algorithms are correct.

Also, we conduct the unified intersection algorithms using massive data; the intersection results of different types are shown, respectively, in Figure 4.

Through several experiments, the unified algorithms proposed by us prove to be correct with regard to the intersection results. However, the time consumption is not so satisfactory. To improve the efficiency and widen the applications of the algorithms, we need to make further research.



FIGURE 3: Results resolved by unified spatial intersection algorithms.

3.2. Parallel Computing Using the Unified Intersection Algorithms in MapReduce. The introduction of two extra dimensions in the unified intersection algorithms brings heavy computation work and results in low efficiency. Furthermore, due to the explosive growth in the volume of geospatial data, many researchers have described the processing of massive data volumes using MapReduce. Zhang et al. implemented parallel spatial querying with a geographical spatial semantic network [34]. Aridhi et al. applied parallel computing to shortest path computing in a large-scale network [35]. The multidimensional-unified representation and computation facilitate parallel computing. Here, we use MapReduce [36] as the programming model to implement parallel computing as in Figure 5.

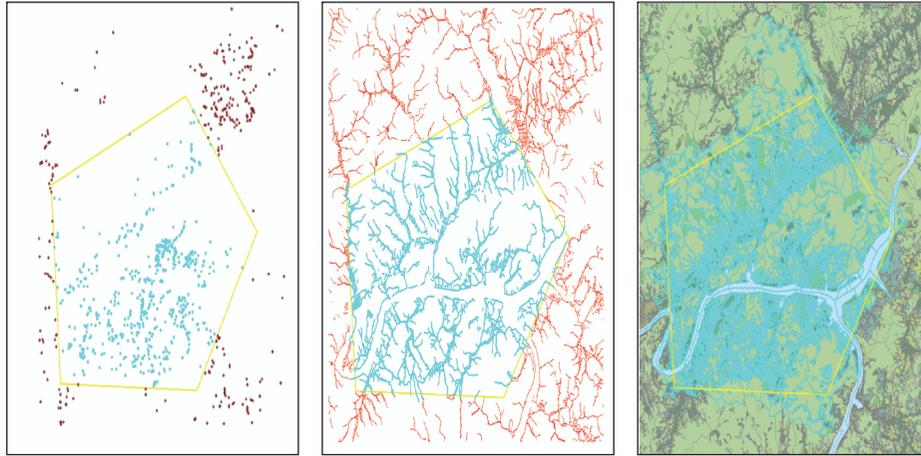


FIGURE 4: Results conducted by unified algorithms using massive data.

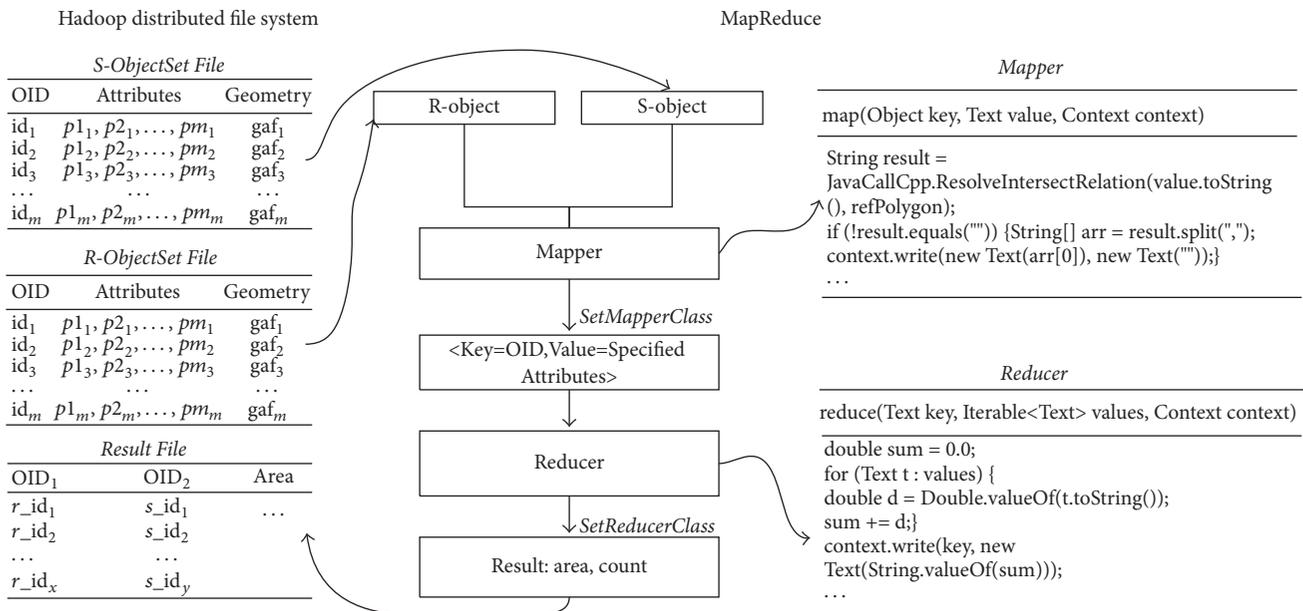


FIGURE 5: Overall parallel computation process.

Hadoop Distributed File System is used to store data in GA structure; we design methods to resolve data into GA structure using open source GDAL/OGR. The attributes and geometry information can be obtained in the method. The intersection algorithms (see Figure 5, named *ResolveIntersectRelation*) are implemented based on GA sandbox (C++), geometric product, and basic operators. We use JNI (Java Native Interface, which can be used to call C++ algorithm in Java environment) to call intersection algorithms in MapReduce. The desired computation results are obtained according to different experiment requirements. For example, we can obtain all objects that intersect with specified objects and sum up the area of the intersecting objects according to some attributes and so on.

We are often confronted with the intersection computing among them. The demand of obtaining the objects that

intersect with the specified spatial extent in land use data is in existence. The traditional method to meet the analogous demands is not unified. In our experiments, we implement the intersection algorithms with the unified algorithms and the specific judgment rules.

In the experiments, we use four computers with the Debian operation system and Hadoop 2.6.0, in which one computer is the namenode and the other three are datanodes. For each computer, the CPU is an Intel® Core™ i-7 4790 CPU@3.60 GHz; there are also 7883 MB of RAM and the hard disk's capacity is 1.0 TB.

Based on the vector data model described above, we organize the geographic objects in a specific CGA format as shown in Algorithm 1.

Finally, the intersection relationships are computed for the four types of different experiment data in the Hadoop

```

<Polygon>
[Polygon, PolygonID, PolygonObjectFormat, Attr1, Attr2, ..., TimeStamp]<Polygon_constraint>
[MBC, m_c[0], m_c[1], m_c[2], m_c[3], m_c[4], m_c[5], m_c[6], m_c[7], m_c[8], m_c[9]]
[Polyline, PolylineID, PolylineObjectFormat, Attr1, Attr2, ..., TimeStamp]<Polyline_constraint>
...
[Point, PointID, PointObjectFormat, Attr1, Attr2, ..., TimeStamp]<Point_constraint>
...
</Polygon>
<Polyline>
[Polyline, PolylineID, PolylineObjectFormat, Attr1, Attr2, ..., TimeStamp]<Polyline_constraint>
[Point, PointID, PointObjectFormat, Attr1, Attr2, ..., TimeStamp]<Point_constraint>
...
</Polyline>
<Point>
[Point, PointID, PointObjectFormat, Attr1, Attr2, ..., TimeStamp]<Point_constraint>
...
</Point>

```

ALGORITHM 1: Data organization of objects.

TABLE 2: Time consumption in MapReduce framework.

Time	Number				
	22183	174881	695343	1386987	5547948
Time 1	12 s	11 s	1 min 23 s	4 min 34 s	23 min 27 s
Time 2	12 s	11 s	1 min 23 s	4 min 35 s	23 min 28 s
Time 3	12 s	11 s	1 min 22 s	4 min 33 s	23 min 29 s

TABLE 3: Time consumption in standalone mode.

Time	Number				
	22183	174881	695343	1386987	5547948
Time 1	6 s	9 s	3 min 58 s	16 min 57 s	No response
Time 2	5 s	9 s	4 min 16 s	15 min 57 s	No response
Time 3	5 s	8 s	4 min 23 s	15 min 21 s	No response

platform and standalone mode, respectively. Each experiment has been done three times to reduce some accident errors. The data in parallel experiments are land use polygons which contain necessary attributes and geometrical information, and the exact amounts are shown in Tables 2 and 3. In the MapReduce frameworks, the computational results obtained are shown in Table 2.

The time consumption results in standalone mode are shown in Table 3. From Table 3, we know that when the number of computed objects exceeds a certain amount, the computing becomes unavailable under specific hardware conditions.

Experiments show that unified intersection algorithms based on CGA can work. The results indicate that the algorithms we present are valid compared to the traditional method. Furthermore, parallel intersection computing using MapReduce is efficient. There are no major advantages in terms of efficiency for using MapReduce or not. However, the

latter one can address the limitations of the standalone mode in processing massive volumes of data.

4. Conclusions

CGA is a promising mathematical theory for representing geographic objects, especially complex and multidimensional geographic objects. This data representation model can be readily extended from low to high dimensions using geometric product. Unified intersection algorithms and specific rules are presented to determine intersection relationships between two geometric objects of different types. The unified representations and intersection algorithms lead to heavy computation; the parallel computation becomes sensible. The resolution of intersection relationships is important for determining spatial relationships. In this study, we investigate the use of an intersection algorithm based on the geometric product. We propose the use of the minimum bounding circle to preprocess the intersection relationship between two objects. This process can improve the efficiency of retrieval and computation. We employ MapReduce frameworks to construct the computing platform for massive data. Experiments show that unified spatial intersection algorithms work well and make up for the shortcomings caused by extra computation of two more dimensions. We can easily identify the objects of different types which meet spatial intersection with specified geometrical objects from massive data. In many domains, for land resource as an example, the intersection computation is needed; we need to compute the objects from land use data intersecting with other land data to make spatial and statistical analysis. In the future, we will extend our intersection algorithm to other spatial relationship algorithms and different domains. The parallel computation strategy is commonly used for massive data, from algorithm to data unit. We partition the data into parallel unit to accelerate our computation. Gaalop [29] is a more simple and efficient GA computation engine for parallel computing, and we look forward to using it in further research.

Competing Interests

The authors declare that they have no competing interests.

Acknowledgments

This research was funded by the National Natural Science Foundation of China (nos. 41471313 and 41671391), the Science and Technology Project of Zhejiang Province (2014C33G20), and the Public Science and Technology Research Funds' Projects (2015418003).

References

- [1] M. F. Goodchild, M. Yuan, and T. J. Cova, "Towards a general theory of geographic representation in GIS," *International Journal of Geographical Information Science*, vol. 21, no. 3, pp. 239–260, 2007.
- [2] Y. Liu, M. F. Goodchild, Q. Guo, Y. Tian, and L. Wu, "Towards a general field model and its order in GIS," *International Journal of Geographical Information Science*, vol. 22, no. 6, pp. 623–643, 2008.
- [3] D. Li and X. Qian, "A brief introduction of data management for volunteered geographic information," *Geomatics and Information Science of Wuhan University*, vol. 35, no. 4, pp. 379–383, 2010.
- [4] Y. B. Yang, C. Q. Cheng, and S. H. Song, "A research on multi-scale expression oriented subdivisional encoding method," *Geography and Geo-Information Science*, vol. 26, no. 5, pp. 33–36, 2010.
- [5] L. Yuan, Z. Yu, S. Chen, W. Luo, Y. Wang, and G. Lü, "CAUSTA: clifford algebra-based unified spatio-temporal analysis," *Transactions in GIS*, vol. 14, no. 1, pp. 59–83, 2010.
- [6] H. B. Li, "From geometric algebras to advanced invariant computing," *Journal of Systems Science and Mathematical Sciences. Xitong Kexue yu Shuxue*, vol. 28, no. 8, pp. 915–929, 2008.
- [7] C. Perwass, *Geometric Algebra with Applications in Engineering*, vol. 4, Springer, Berlin, Germany, 2009.
- [8] C. Perwass, "Teaching geometric algebra with CLUCalc," in *Proceedings of the International Symposium on Innovative Teaching of Mathematics with Geometric Algebra*, pp. 33–50, Kyoto, Japan, 2003.
- [9] R. Ablamowicz and B. Fauser, "Mathematics of Clifford: a Maple package for Clifford and Grassmann algebras," *Advances in Applied Clifford Algebras*, vol. 15, no. 2, pp. 157–181, 2005.
- [10] D. Fontijne, "Gaigen 2: a geometric algebra implementation generator," in *Proceedings of the 5th International Conference on Generative Programming and Component Engineering (GPCE '06)*, pp. 141–150, Portland, Ore, USA, October 2006.
- [11] D. Hildenbrand and E. Hitzer, "Analysis of point clouds using conformal geometric algebra," in *Proceedings of the 3rd International Conference on Computer Graphics Theory and Applications (GRAPP '08)*, pp. 99–106, Funchal, Portugal, January 2008.
- [12] C. Schwinn, A. Görlitz, and D. Hildenbrand, "Geometric algebra computing on the CUDA platform," in *Proceedings of the 1st International Workshop on Computer Graphics, Computer Vision and Mathematics (GraVisMa '09)*, pp. 111–117, September 2009.
- [13] L. W. Yuan, G. N. Lü, W. Luo, Z. Y. Yu, L. Yi, and Y. H. Sheng, "Geometric algebra method for multidimensionally-unified GIS computation," *Chinese Science Bulletin*, vol. 57, no. 7, pp. 802–811, 2012.
- [14] Z. Y. Yu, W. Luo, Y. Hu, L. W. Yuan, A. X. Zhu, and G. N. Lü, "Geometric algebra model for geometry-oriented topological relation computation," *Transactions in GIS*, vol. 20, no. 2, pp. 259–279, 2015.
- [15] Y. Hu, W. Luo, Z. Y. Yu, L. W. Yuan, and G. N. Lü, "Geometric algebra-based modeling and analysis for multi-layer, multi-temporal geographic data," *Advances in Applied Clifford Algebras*, vol. 26, no. 1, pp. 151–168, 2016.
- [16] W. Luo, Y. Hu, Z. Y. Yu, L. W. Yuan, and G. N. Lü, "A hierarchical representation and computation scheme of arbitrary-dimensional geometrical primitives based on CGA," *Advances in Applied Clifford Algebras*, 2016.
- [17] Z. Yu, W. Luo, Y. Hu, L. Yuan, A.-X. Zhu, and G. Lü, "Change detection for 3D vector data: a CGA-based Delaunay-TIN intersection approach," *International Journal of Geographical Information Science*, vol. 29, no. 12, pp. 2328–2347, 2015.
- [18] T. Möller, "A fast triangle-triangle intersection test," *Journal of Graphics Tools*, vol. 2, no. 2, pp. 25–30, 1997.
- [19] M. Held, "ERIT—a collection of efficient and reliable intersection tests," *Journal of Graphics Tools*, vol. 2, no. 4, pp. 25–44, 1997.
- [20] O. Tropp, A. Tal, and I. Shimshoni, "A fast triangle to triangle intersection test for collision detection," *Computer Animation & Virtual Worlds*, vol. 17, no. 5, pp. 527–535, 2006.
- [21] P. Guigue and O. Devillers, "Fast and robust triangle-triangle overlap test using orientation predicates," *Journal of Graphics Tools*, vol. 8, no. 1, pp. 25–42, 2003.
- [22] Z. Zong, L. Yuan, W. Luo, Z. Yu, and Y. Hu, "Triangulation intersection algorithm based on conformal geometric algebra," *Acta Geodaetica et Cartographica Sinica*, vol. 43, no. 2, pp. 200–207, 2014.
- [23] J. B. Shen, J. H. Li, T. C. Kong, and D. B. Li, "Fast intersection test algorithm of spatial convex polygons," *Journal of Daqing Petroleum Institute*, vol. 32, no. 1, pp. 80–82, 2008.
- [24] L.-Y. Wei, "A faster triangle-to-triangle intersection test algorithm," *Computer Animation and Virtual Worlds*, vol. 25, no. 5–6, pp. 553–559, 2014.
- [25] K. Sugihara, "An intersection algorithm based on delaunay triangulation," *IEEE Computer Graphic & Application*, vol. 12, no. 2, pp. 59–67, 1992.
- [26] Q. Zhou, E. Zhong, and Y. Huang, "A parallel line segment intersection strategy based on uniform grids," *Geo-Spatial Information Science*, vol. 12, no. 4, pp. 257–264, 2009.
- [27] F. Wang, "A parallel intersection algorithm for vector polygon overlay," *IEEE Computer Graphics and Applications*, vol. 13, no. 2, pp. 74–81, 1993.
- [28] L. Dorst, D. Fontijne, and S. Mann, *Geometric Algebra for Computer Science: An Object-Oriented Approach to Geometry*, Morgan Kaufmann Publishers, San Francisco, Calif, USA, 2007.
- [29] D. Hildenbrand, *Foundations of Geometric Algebra Computing*, vol. 8, Springer, Heidelberg, Germany, 2013.
- [30] J. Zamora-Esquivel, " $G_{6,3}$ geometric algebra; description and implementation," *Advances in Applied Clifford Algebras*, vol. 24, no. 2, pp. 493–514, 2014.
- [31] E. Hitzer, "The geometric product and derived products," 2003, <http://vixra.org/pdf/1306.0175v1.pdf>.
- [32] D. Hestenes and G. Sobczyk, "Clifford algebra to geometric calculus," *Fundamental Theories of Physics*, vol. 5, no. 69, 1984.
- [33] J. Cameron and J. Lasenby, "Oriented conformal geometric algebra," *Advances in Applied Clifford Algebras*, vol. 18, no. 3–4, pp. 523–538, 2008.

- [34] C. Zhang, T. Zhao, L. Anselin, W. Li, and K. Chen, "A MapReduce based parallel approach for improving query performance in a geospatial semantic web for disaster response," *Earth Science Informatics*, vol. 8, no. 3, pp. 499–509, 2014.
- [35] S. Aridhi, P. Lacomme, L. Ren, and B. Vincent, "A MapReduce-based approach for shortest path problem in large-scale networks," *Engineering Applications of Artificial Intelligence*, vol. 41, pp. 151–165, 2015.
- [36] J. Dean and S. Ghemawat, "Map Reduce: a flexible data processing tool," *Communications of the ACM*, vol. 53, no. 1, pp. 72–77, 2010.



Hindawi

Submit your manuscripts at
<http://www.hindawi.com>

