

Shape computing: A new geometric computing mechanism

Yuanjun He¹, Haiyan Yu², Hongming Cai¹, Zhenghong Peng³, Wei Liu¹,
Zhiping Hu⁴

¹*Shanghai Jiao Tong University, Shanghai, China*
yjhe@sjtu.edu.cn, hmcai@sjtu.edu.cn, liu-wei@sjtu.edu.cn

²*Donghua University, Shanghai, China*
yuhy@dhu.edu.cn

³*Wuhan University, Wuhan, Hubei, China*
laopeng129@vip.sina.com

⁴*Shanghai Customs College, Shanghai, China*
zhiping_hu@163.com

Abstract. Graph/image has become an important computing source, object and representing result, and it is also pursued as a representing form for the solution. The processing work of graph/image is mainly geometric computing. A geometrized computing mechanism named “Shape Computing” is proposed. It considers the representation of shape and the generation of graph more from the perspective of geometry itself, which is more conducive to the processing of geometric relations. Hence the algorithm framework can be built from a more macroscopic view, and the computing process is more structured, intuitive and simplified. The general ideas and main strategies of this mechanism are expounded, the framework and implementing strategies are then constructed, application examples along with comparative analysis of computing performance are also given. Compared with current “numeric computing”, Shape Computing is better designed for the fusion of multi-dimensional spaces of “three-dimensional conceptual thinking, two-dimensional graphic construction, one-dimensional numerical calculation”. Theoretical analysis and application examples demonstrate that the proposed Shape Computing contributes to improve the readability, to reduce computational complexity, and to deal with computing robustness.

Key Words: Shape Computing, geometry, geometric computing, algebraic computing, graphic construction

MSC 2020: 68U05

1 Introduction

The basic task in graphics is the transformation from shape to graph and from graph to shape, wherein, the essence is geometric representation and geometric computation, and the theoretical foundation is geometry. As James R. Miller said [6], “Computer graphics and modeling rely on mathematical operations on points and vectors. I would advocate using vector geometric analysis to simplify required derivations.” However, the current computing architecture relies on algebra dominated computing mechanism (numerical computing for short in this paper), based on floating point numbers under John von Neumann binary system. Therefore, graphics computing has to be transformed into complicated algebraic representations and calculations. With the increasing computing scale and complexity of the problems in graphics, the limitations of this computational mechanism have become more and more obvious.

Robustness. Two main reasons cause un-robustness in geometric computation. One is digital calculating errors, relating to the accumulation of digital and calculation errors, especially the floating-point errors. The other is the geometry itself resulting from geometric singularity as bounded models lead to various special geometric relationship (co-point, co-line, co-plane). The judgment of geometric singularity is usually uncertain. Consequently, the impact from geometric singularity is fundamental for the robustness of geometric computing. Currently, there is no effective method to judge geometric singularity. Christer Ericson [11] once expressed his concern about those researches that only focus on speed and ignore robustness, and he believes that large-scale random tests have difficulties to detect conditions that affect the robustness of an algorithm.

Some defects in numerical computing have to be left aside. In numerical computing, the solving process is often involved with a large number of complex algebraic formulas, which are difficult for a human to understand and communicate. The nonintuitive way degrades people’s spatial thinking, and geometric computing often becomes unreadable or even uncontrollable. Stephen Hawking famously wrote [3], “some told me that each equation I included in the book would halve the sales. I therefore resolved not to have any equation at all.” This shows the lethality of the unreadability of algebra to human communications.

With the expansion of applications in CAD system, intelligent manufacturing, and machine vision, the demand and the requirements for complex computing and precise computing are increasing, especially in industrial software, which poses new challenges to geometric computing [10]. As a result, problems of un-robustness, uncontrollability, difficulty in communication etc., caused by the limitations of numerical computing, need to be concerned and to be resolved.

This paper thus proposes a new computing mechanism named “Shape Computing Mechanism” (Shape Computing for short) to geometrize graphics computing. It seeks a global and intuitive solution from a geometric perspective, and assists to numeric computing mechanism, which will largely alleviate the unreadability and un-robustness caused by geometric singularity, and reduce computational complexity.

2 Frameworks of Shape Computing

2.1 The Concept

There are mainly two kinds of reasoning in mathematics: symbolic reasoning and intuitive reasoning. The former originates from numeric system while the latter does from the shape

system. After “number”, “shape” has been introduced as the second main concept of mathematics. The imageability, intuition, accuracy and conciseness of graphics/images give full play to people’s advantage in spatial thinking, and through them, people understand the unknown and explore the truth. After computers became the main computing tool, the computing methods and the representation of solutions have undergone revolutionary changes. However, all changes are inseparable from their origin, and the two main methods remain unchanged in representing computing objects and results: number and shape.

Computers have advantages on numerical operations over humans, whereas humans are better at graphic thinking. Euclid systematically employed graphics in his book *The Elements*, with a small number of symbols and a large amount of logic. His great work combines two innovations: The use of graphics and the logical structure for proof. Monge’s *Descriptive Geometry* represented and computing geometric problems in a non-algebraic way.

In geometry, a problem is considered from the 3D space and it is in line with people’s spatial cognitive thinking. Algebra involves the operation of time where a problem is solved in one dimensional order and gives full play to computer’s calculating power. The biggest problem of a computing mechanism relying solely on numeric computing is that people’s spatial thinking is detached by such one-dimensional numerical computing.

The proposed Shape Computing aims to constructing a buffer between graphic thinking and numeric computing (Figure 1). In the level of computing design, Shape Computing designs an algorithm at the spatial level, based on the consideration of shape as a whole, where human is in a more dominant position and leading role. In the level of computing implementation, basic geometric operations are taken as computing cell, where the numeric calculations are decomposed and encapsulated. Only in the end, the dull numerical calculations are left to computer process. Thus, people are oriented to 3D geometric space, while computing is oriented to 1D algebraic operations, which improves transforming mechanism between shape and number to a higher level for comprehensibility. Consequently, Shape Computing defines thinking, geometry, algebra and computation from four different levels: Thinking is on the design level, geometry is on the expression level, algebra is on the computing level, and calculation is on the implementation level [4].

2.2 The Framework

The overall framework of Shape Computing is shown in Figure 2. The computing method in Shape Computing is similar to human’s graphic thinking and the solving process in descriptive geometry. For a spatial geometric problem, the first step is to seek the best view for observation in space. Then the spatial problem is projected onto a plane, and a complex geometric problem is further decomposed into several basic geometric operations by means of graphic construction. Finally, the solution of the 3D problem is obtained by inverse construction. Thus, in the whole computing process, geometry is taken as the computing unit, and algebraic calculation is desalinated. In the level of shape-number interaction, a switching variable is defined to assist the expression of geometric relations. The computing method in Shape Computing is similar to human’s graphic thinking and the solving process in descriptive geometry. For a spatial geometric problem, the first step is to seek the best view for observation in space. Then the spatial problem is projected onto a plane, and a complex geometric problem is further decomposed into several basic geometric operations by means of graphic construction. Finally, the solution of the 3D problem is obtained by inverse construction. Thus, in the whole computing process, geometry is taken as the computing unit,

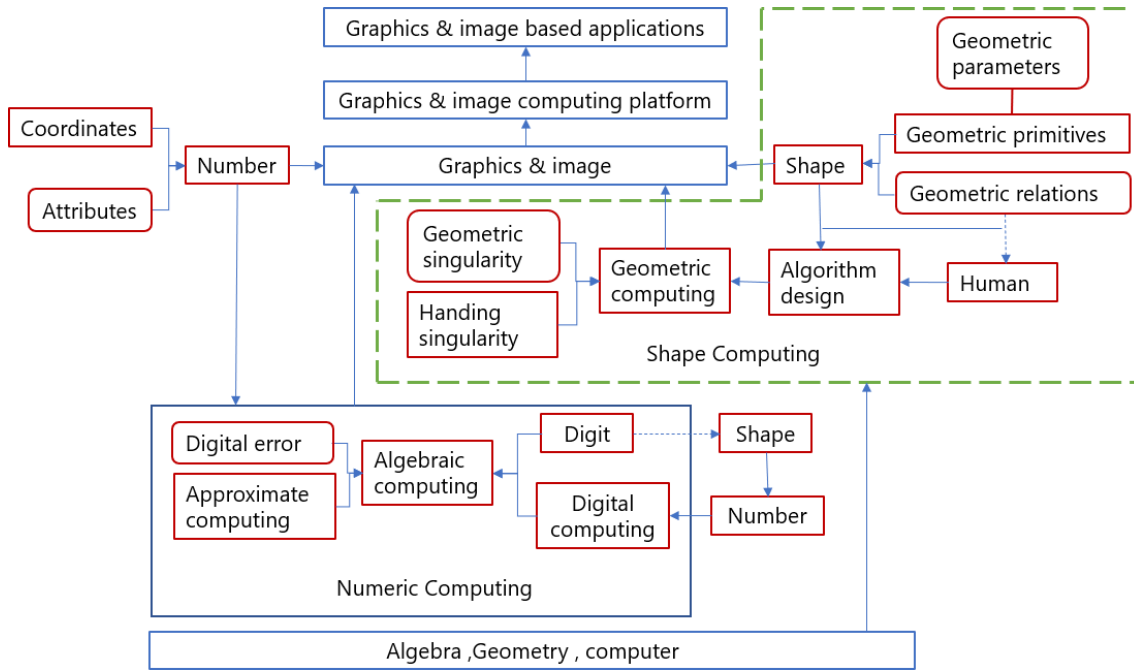


Figure 1: The orientation of Shape Computing

and algebraic calculation is desalinated. In the level of shape-number interaction, a switching variable is defined to assist the expression of geometric relations.

The following new computing elements and strategies are introduced to construct the unified and robust geometric computing mechanism.

- ① **Geometric Number (GN).** GN adopts the theory of vector geometry to define geometric directions and relations. It simplifies the computing processes and the selection of solutions, and assists the whole computing process. GN originated from the theory of yin and yang in China and the theory of 0/1 in computers.
- ② **Geometric Base (GB).** GB is the basic computing unit in Shape Computing. The consequence of GB represents not only the process of geometric construction, but also the solution. GB absorbs the idea of ruler-gauge drawing in descriptive geometry, the definition of the base of a vector in linear algebra and concept of algorithms in computer science.
- ③ **Geometrized Transform (GT).** Vectors or normal directions of planes are used to construct the elements of transforming matrix. And then the geometric transformation, the definition and solution functions for basic geometries are both unified. As a result, all transformations are uniformly described, such as translation, rotation, staggered tangent, symmetry and proportion.
- ④ **Solving Strategy of Planar Problems.** The geometric problems are decomposed level by level until to the most basic geometric relations. The data structure of constructing (solving) tree is therefore established, and through traversing the tree, the geometric solution is represented by a sequence of GB (which is also a new GB).
- ⑤ **Solving Strategy of Spatial Problems.** A computing coordinate system is established by taking the main geometric elements as the reference. A spatial problem is then reduced to a planar one through the proposed algorithm of projecting to an arbitrary plane. Finally, the solution of the spatial problem is obtained by reversing the con-

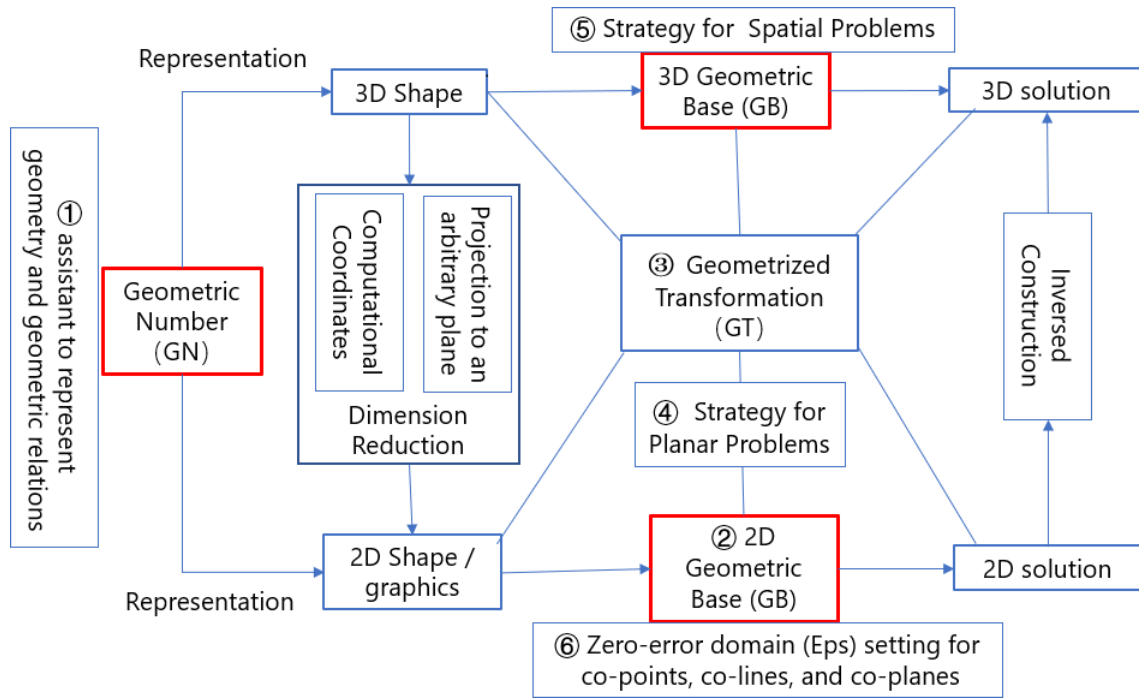


Figure 2: The overall frame and solution mechanism of Shape Computing

struction of the planar solution.

- ⑥ **Zero-error Domain Setting (ZD)**. Given the floating-point number system in computers, the zero-error domain is set to meet precision requirement in different engineering fields, beside setting the floating-point type in double precision. Within the range of zero domain, two points, two lines and two planes are supposed to coincide.

By employing the above strategies, a whole set of theories and methods are further constructed for the solution of un-robustness caused by geometric singularity. In particular, the geometric singular problems are considered at the geometric level and solved by simple operations of GN in intersection points.

3 Key Points of Shape Computing

Computationalists believe that the essence of life does not lie in specific matter, but in the form of organization of elements; so as long as elements can be constructed in the appropriate form, this new system can manifest life [8]. Similarly, the key to geometric computing is the computation of geometric relations, including the complete representation, calculation and reconstruction. This section will give the key points for the computation of geometric relations, including GN-aided representation, GB-based algorithm design, geometrized and uniformed transformation, and computation through dimension reduction.

3.1 GN-aided Representation of Geometric Relations

Developing a geometric language to represent and compute graphic problems is the goal that scientists such as Leibniz have been pursuing. Leibniz proposed the idea of “constructing a geometric number” as a geometric calculation unit [1]. In shape computing, GN assists the definition of geometries and relations. It is a vital bond which unites the entire computing

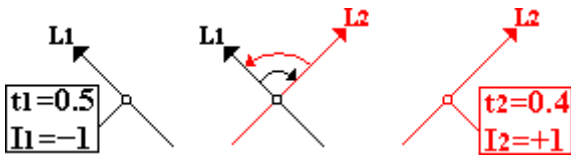


Figure 3: The definition of GNIP (t_1 and t_2 are the positional parameters of the intersection point on L_1 and L_2)

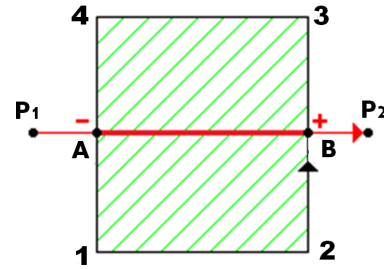


Figure 4: The portion of vector P_1P_2 between negative GNIP (point A) and positive GNIP (point B) is inside the shadowed graph.

process of geometric definition, representation, measurement and operation with an aid to simplify computing process and the choice of solutions.

The main services of GN are to distinguish geometric relations with switching properties, including: (1) direction of a geometric primitive such as a straight line, circle/arc, surface, etc.; (2) entry/exit situation of an intersection point; (3) outer/inner (left/right) situation of a boundary; (4) direction of a geometric connection following pulley rule; (5) direction of a unit normal vector of a line, plane, etc.; (6) positive/negative situations of geometric measurement of length, area, volume, etc.

The mathematic foundation of GN in Shape Computing is vector geometry. The definition and function of GN will be given with the example of GN for intersecting points (GNIP for short), which is the mostly used GN in geometric computing.

3.1.1 The Definition of GNIP

Suppose that two vectors L_1 and L_2 intersect. Rotate L_1 around the intersecting point and make it coincide with L_2 . If the direction of rotation is clockwise (Figure 3), then the GNIP relative to vector L_1 is -1 (denoted as $I_1 = -1$), and the GNIP relative to the vector L_2 is $+1$ (denoted as $I_2 = +1$).

As mentioned above, geometric computing is mainly the computation of geometric relations, and the key is to distinguish inner and outer between geometries. A simple application of GNIP is given to illustrate the basic theory (Figure 4). To determine the portion of P_1P_2 inside the boundary 12341, two intersecting points have to be solved, wherein the focus is “the relative properties of the position” such as the entry-point and the exit-point. According to the definition of GNIP, for P_1P_2 , $IA = -1$ (the GNIP of point A), and $IB = +1$ (the GNIP for point B). Thus, it can be determined that P_1P_2 enters the boundary at point A, and exits the boundary at point B. It is often not the main focus in computation that point A is on boundary 41 and B is on boundary 23.

The method of using GN to represent the properties of “in and out” for intersecting points has widely applications in hatching drawing, polygon clipping, and Boolean operations. Figures 5 and 6 are more typical and complex examples of applying GN to assist to deal with singular intersections. Figure 12 further shows how GNIP works for the construction of geometric relationship in Boolean operations.

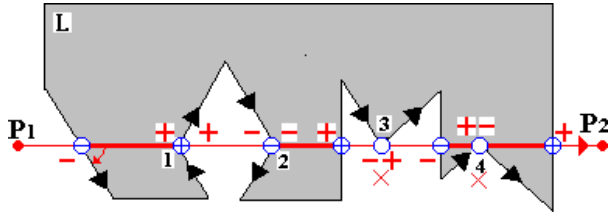


Figure 5: Processing rules of overlapping points

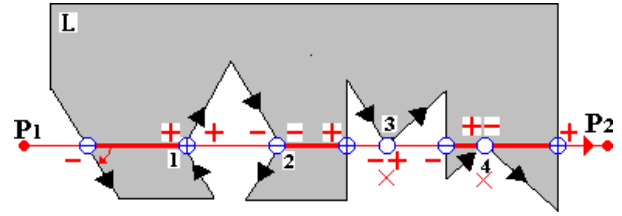


Figure 6: Processing rules of overlapping edges

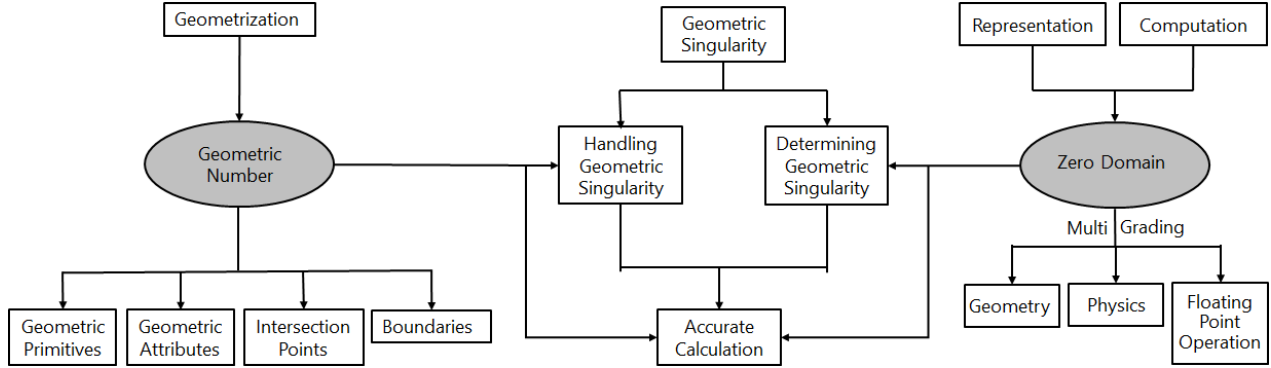


Figure 7: The overall solution to the geometric singular in Shape Computing

3.1.2 Handling Overlapping Intersection Points and Edges

Two rules are given to handle geometric singularity with simple operations of the GNIP. Furthermore, this solution is built on a theoretical level.

The rule for overlapping intersection points (Figure 5). Accumulate the values of GN of overlapped intersecting points. If the algebraic sum is zero, then delete these points (3, 4 in Figure 5); otherwise, combine them into one point (1, 2 in Figure 5), and the GN of the combined point is defined by the sign of the algebraic sum.

The rule for intersection points in the same vector (Figure 6). When two consecutive intersection points are on the same vector and their values of GN are the same (7 and 8 in Figure 6), if the values of GN are both “ ± 1 ”, then delete the second intersection point (7 in Figure 6); if the values of GN are both “ -1 ”, then delete the first intersection point (8 in Figure 6).

Thus, through the simple addition operation of GN (the value is $+1$ or -1), geometric singularity caused by overlapped points and edges are dealt with.

3.1.3 Scheme for Singular Problems in Geometric Computing

On the basis of the above analysis, the scheme for the geometric singular problem is given (Figure 7), generally including the following four procedures.

- Assign the value of GN for the intersection point.
- Zero Domain. In the geometric computing system, the unified Zero-domain (ε) is established and co-point precision is adjusted to adapt to the accuracy requirements in different application fields. For example, in the field of ship building, the allowed error

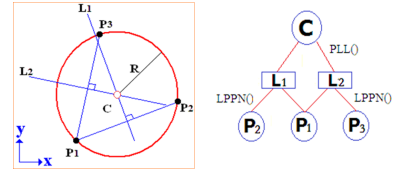
Algorithm 1: Solving and representing a planar circle through three points with a sequence of GB; the sequence also constructs a new GB defined by `cppp()`=`lpp()`; `lppn()`; `p1l()`; `dpp()`.

Process	The sequence of GB
1) Construct the perpendicular bisector of P_1P_2 , and get L_1 ; (The situation of coinciding points P_1 and P_2 is detected.)	GB: <code>lppn()</code>
2) Construct the perpendicular bisector of P_1P_3 , and get L_2 ; (The situation of coinciding points P_1 and P_3 is detected.)	GB: <code>lppn()</code>
3) Find the intersecting points of L_1 and L_2 and get the center point C ; (The situation of three points in the same line is detected.)	GB: <code>p1l()</code>
4) Calculate the radius R ; (The distance from the center to one of the three points is R .)	GB: <code>dpp()</code>

$$y_c = \frac{[(x_2^2 - x_1^2) + (y_2^2 - y_1^2)](x_3 - x_2) - [(x_3^2 - x_2^2) + (y_3^2 - y_2^2)](x_2 - x_1)}{2[(y_2 - y_1)(x_3 - x_2) - (y_3 - y_2)(x_2 - x_1)]}$$

$$x_c = \frac{(x_2^2 - x_1^2) + (y_2^2 - y_1^2) - 2y_c(y_2 - y_1)}{2(x_2 - x_1)}$$

a) algebraic method



b) geometric method

Figure 8: Two methods to solve and represent a circle through three points

is usually in the millimeter range, while in the field of integrated circuit, the allowed error is usually in micrometer range.

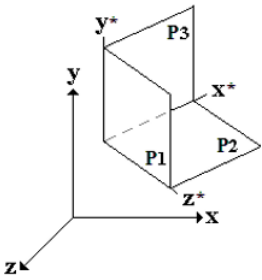
- Detection of singularity. If two points P_1 and P_2 meet the condition “ $|P_1P_2| < \varepsilon$ ”, then they are co-points. These two points are in singular situation.
- Treatment of singularity: The singular situations of co-points, co-lines, co-planes etc., are culled based on the rules of handling overlapped intersection points and edges.

3.2 GB-based Computing Algorithm

In descriptive geometry, a few basic drafting operations with ruler and gauge construct complex engineering drawings. In advanced algebra, any vector in a linear space can be expressed linearly by basis vectors. In computers, a complex algorithm can be constructed by a sequence of simple and independent algorithms. Based on these ideas, in Shape Computing, GB is introduced as the computational base. Then the solution of a geometric problem can be represented by a sequence of GB.

Taking as an example the computation of a circle through three points P_1 , P_2 and P_3 , the solving strategy and the representation of the computing result based on GB are given in Algorithm 1 and are compared with algebraic method in Figure 8.

The comparison shows that in this example, the algebraic functions are difficult to understand. Shape Computing decomposes a geometric problem to the most basic geometric relations level by level, and the construction (solving) tree is therefore established. Finally, the solution is represented with a sequence of GB by traversing the tree. Consequently, this proposed method is intuitive and clear.

$$\begin{cases} P_1(x^* = 0): a_1x + b_1y + c_1z + d_1 = 0 \\ P_2(y^* = 0): a_2x + b_2y + c_2z + d_2 = 0 \\ P_3(z^* = 0): a_3x + b_3y + c_3z + d_3 = 0 \end{cases}$$


$$(X^*, Y^*, Z^*, H^*) = (x, y, z, 1) \cdot \begin{pmatrix} a_1 & a_2 & a_3 & 0 \\ b_1 & b_2 & b_3 & 0 \\ c_1 & c_2 & c_3 & 0 \\ d_1 & d_2 & d_3 & 1 \end{pmatrix} = (x, y, z, 1) \mathbf{T}_{xyz \rightarrow x^*y^*z^*}$$

Figure 9: Constructing new coordinate system from normal vectors of three intersecting planes

3.3 Geometrized Transformation

From the aspect of geometry, a new planar coordinate system can be constructed with unit normal vectors of any two intersecting non-collinear lines (spatial coordinates with three such planes), and the transformation matrix between the new and old coordinates are defined by the normal coefficients of intersecting normal vectors in the homogeneous form (Figure 9). This is the geometrized description of the transformation, which is more intuitive and unified than the algebraic description with transformation matrix obtained by translation, rotation, etc. [9].

In Figure 9, in the coordinate system xyz , there are three spatial intersecting planes P_1 , P_2 and P_3 which have common intersection point. Their normal vectors form a new spatial coordinate system $x^*y^*z^*$. Then the transformation matrix for any spatial point, from xyz to $x^*y^*z^*$ (and the inverse transformation matrix) can be represented by the equation coefficients of the three intersecting planes (the normal coefficients of the planes and the homogeneous items).

3.4 Computation Through Dimension Reduction

Dimension Reduction is an effective method of reducing geometric complexity. Projection in descriptive geometry is a kind of dimension reduction which transforms a 3D problem to a 2D one and hence to divide and conquer, and further simplify and split a problem. The basic computation strategy based on dimension reduction is given below (Figure 10) [9].

First, a local computational coordinates system is constructed by taking the main geometric element as the main reference. Then the spatial problem is reduced to a planar one based on the method of projecting to an arbitrary plane. Finally, the spatial solution is obtained by inverted construction with planar solutions. This method is beneficial to reduce the computational complexity, to simplify the analysis of geometric singularity and to improve the robustness of computing.

4 Cases Study Based on Shape Computing

More than 300 algorithms derived from Shape Computing are provided in reference [4], including basic geometric algorithms, geometric transformations, 2D and 3D computing and

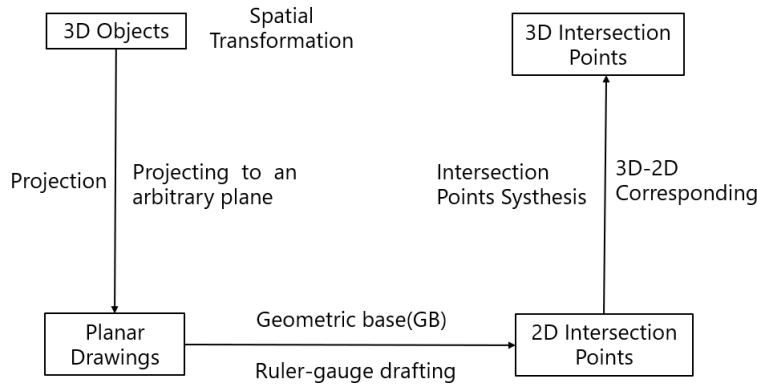


Figure 10: Computation through dimension reduction

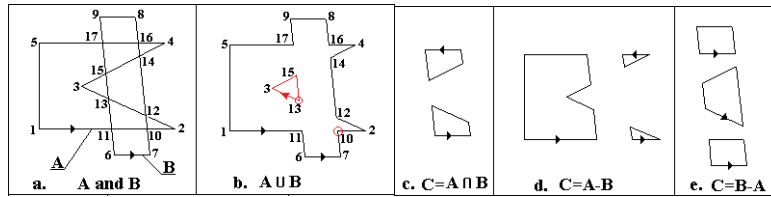
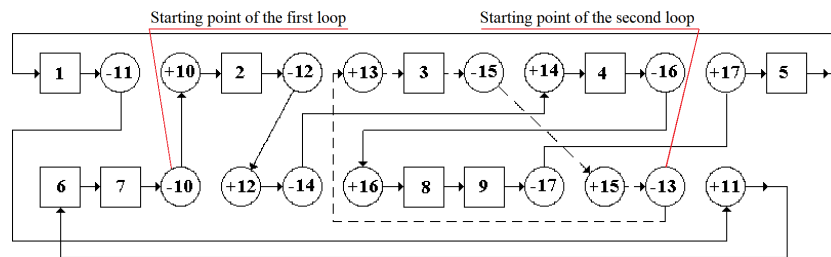


Figure 11: Boolean operations of A and B

modeling. Several typical examples are given to illustrate the implementation and performance of Shape Computing.

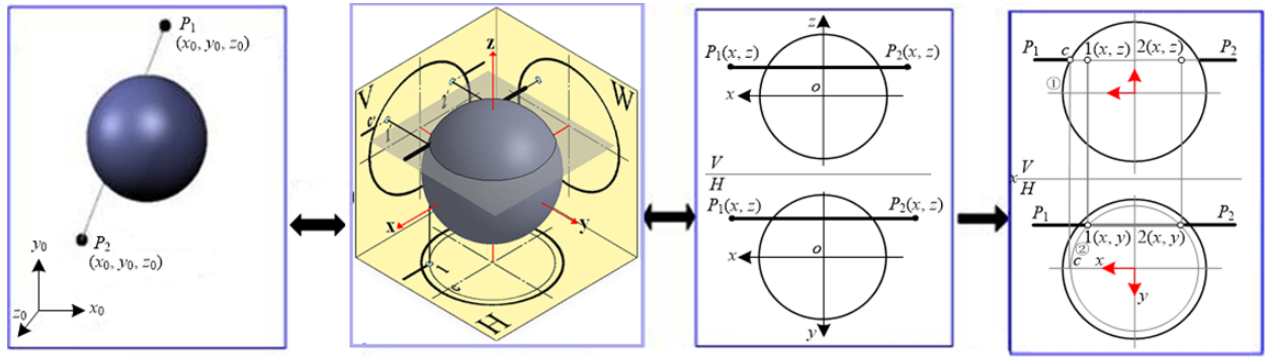
4.1 Boolean Operations Based on GN

Boolean operation is the process to reorganize topologically two boundaries and GN simplifies this topological reorganization with the following procedures (Figure 11). (1) Starting from a certain intersection point, do set operations (such as union, intersection and set difference). (2) Repeat judging until arriving at the first intersection point and a new boundary (loop) is obtained. If GNIP is negative, then turn to another loop; if GNIP is positive, then the vertex is searched in the original direction. (3) Once all intersection points are traversed, the Boolean operation is completed and the algorithm ends. Figure 12 shows the process of union operation of two polygons A and B in Figure 11, and the result is shown in Figure 11.b.



Two boundary loops of $A \cup B$ are obtained starting from intersection 10 and 13, respectively (Figure 11.b). The circled number represents an intersecting point; squared number represents a vertex.

Figure 12: Union operation for two polygons A and B in Figure 11



a) The 3D problem b) Transforming to a CSS c) Dimension reduction d) 2D solution

Figure 13: Implementation of Shape Computing to the intersection of line with sphere in 3D space

4.2 Computation Scheme for Spatial Problems

In Shape Computing, for spatial problems, dimension reduction is used as much as possible, usually based on the theory of projection and 2D/3D corresponding method in descriptive geometry. the procedure of the algorithm of line interacting with sphere in 3D space is shown in Figure 13.

Step1 (preprocessing). Construct a computational coordinate system (CCS for short) taking the simplified representation of primary geometry (in this example, the primary geometry is the sphere) as the main concern and then make transformation (Figure 13.a, b);

Step2 (3D solution). Under CCS, find the intersections on two projection planes (Figure 13.c, d);

Step3 (post-processing). Inversely transform the intersection points to the original 3D space.

In the example, the intersecting points are carried out finally by using twice planar GB of “intersection of a line segment with a circle.”

5 Comparison and Discussion

5.1 The Computational Efficiency

Line clipping algorithms developed from Shape Computing are taken as examples to show the performance of dimension reduction in Shape Computing.

1) Line clipping against a 2D and 3D rectangular window A line clipping algorithm is developed based on dimension reduction in Shape Computing (Figure 14) [4]. The performance was compared with three well known clipping algorithms, i.e., Cohen Sutherland [7] (CS), Cyrus Beck [2] (CB) and Liang Barsky (LB) [5]. Three types of test samples are designs for the comparing tests. (a) 20 2D lines segments that each have two intersections with the window (Figure 15.a); (b) 120 2D lines segments containing inside, outside, intersecting the window and singular positions (Figure 15.b); (c) 123 3D lines segments.

In the performance test, four clipping algorithms were repeated 1 million times. The running time is shown in Table 1. CB is not listed as it is designed for line clipping against

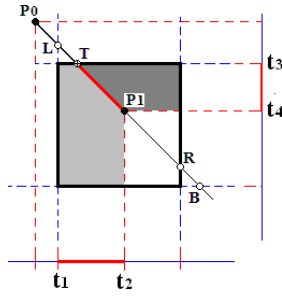


Figure 14: Theory of Shape Computing clipping

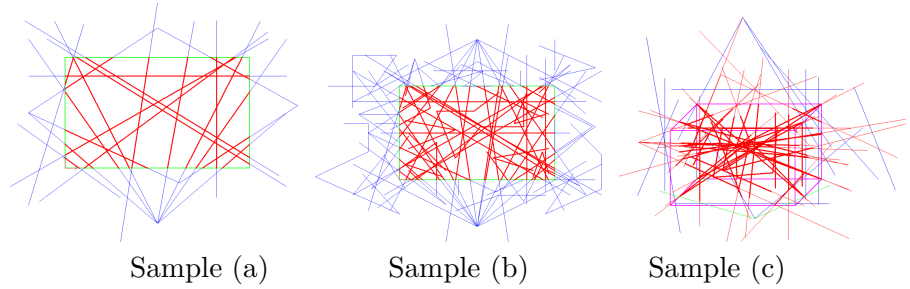


Figure 15: Test samples for comparison of four algorithms

Table 1: Computing efficiency of three clipping algorithms (clipping one million times for each sample)

Clipping algorithm	Sample (a)	Sample (b)	Sample (c)
Cohen Sutherland (CS)	679	3180	7121
Liang Barsky (LB)	603	3231	6758
Shape Computing	544	2596	6236

polygons and hence its speed for a rectangular window is relative slow (its 2D clipping time is 3400 and 19911 for test sample (a) and (b) respectively).

The test shows that our clipping algorithm is slight faster than others, although the computing speed of the three algorithms are in the same order of magnitude. Furthermore, it should be noted that the comparison is between the routine algorithm and specifically designed algorithms. The proposed algorithm is derived from the algorithm framework of Shape Computing. Algorithms of CS and LB are both designed just for the problem of line clipping.

2) Line clipping against a pyramid (frustum clipping) Shape Computing was compared with LB [5] and Line-Surface-Direct Intersection method (DI) using 78 line segments including situations where the vertexes, edges and boundary faces of the clipped frustum are in singular situations (Figure 16). The test result shows that the computing speeds of three algorithms are also on the same order of magnitude, and the reference ratio is [4]:

$$LB : SC : DI = 4243 : 4212 : 4228$$

Consequently, the performance of Shape Computing for frustum clipping can also compete with classical line clipping algorithms.

5.2 Discussion on Shape Computing

Combined with the above examples, we discuss Shape Computing from the aspects of geometric algorithm design, modeling, and execution.

(1) A more macroscopic view for geometric algorithm design Shape Computing focuses on getting a qualitative result by separating boring and repeated algebraic calculation to the computer. Thus, the computing-based solution process is more structured, intuitive and

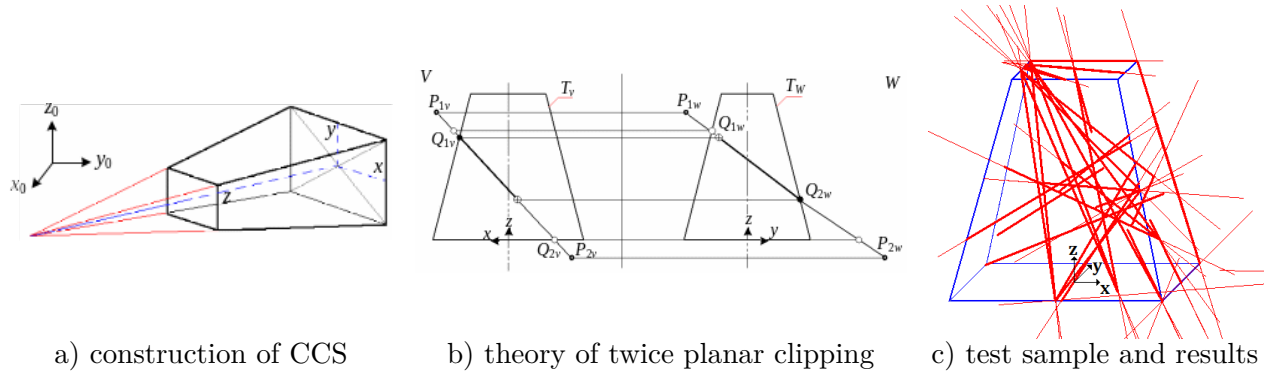


Figure 16: Theory and test results of CS for Frustum Clipping

simplified. Therefore, algorithm design has become a process of “thinking from a qualitative and intuitive view so as to carry out problem solving in a quantitative and orderly way”, reaching the realm of “shape thinking and number calculation.”

Shape Computing solves the problem of un-robustness caused by geometric singularity from the perspective of geometry at the theoretical level. It provides a more appropriate method to judge and solve geometric singularity to overcome the uncertainty.

(2) A higher level of geometric model and relations representation Shape Computing can better represent geometric relations. The basic geometry can be expressed by analytical equations, but the geometric relations are difficult to be fully expressed by the combination of algebraic equations. For example, the relationship between two planar straight-line segments includes separation, point-to-point collision, point line collision, complete coincidence, partial coincidence, etc. In 3D space, geometric relations are more complex. Shape Computing represent geometric relations in higher level of model than numeric computing.

(3) A more interpretability and robustness way for geometric algorithms execution In Shape Computing, geometrized transformation unifies the expression of transformation matrix. Geometric Base constructs the base of geometric solution, and the overall robustness of geometric computation is thus improved.

Therefore, algorithms developed from Shape Computing is much more readable. Algebra-based algorithms always involve a lot of complex algebraic formulas. With the increasing computing scale, speed and complexity, and demand of robustness and accuracy, the algorithms are difficult for people to understand and communicate. Furthermore, the correctness and robustness of algorithms can only be verified by massive tests.

6 Conclusions

Shape Computing solves the following scientific and practical problems. In the conceptual level, it pursues a breakthrough in the combination of shape and number. In the mechanism level, it presents mechanisms of geometrized representation, operation, computing method, solution expression, dimension reduction and geometric transformation. In the target level, it solves some key problems such as dimension gaps and computational robustness, and particularly, a set of theory and solutions is proposed to solve geometric singular problems.

Shape Computing uses the view of geometers to think about problems macroscopically and carefully and solves problems in the view of modern mathematicians strictly and orderly. It considers shape as one object, so that the geometric problems could be solved in a multi-dimensional space from the geometric perspective. Therefore, the algorithm framework is designed at a macro and higher level. In the solving process, the complex numerical calculations are decomposed to basic ones and are put into algebraic implementation in the final step. In whole, the computing process is structured, intuitive and simplified. Shape Computing realizes the multi-dimensional space fusion of “3D conceptual thinking, 2D graphic construction, and 1D numerical calculation”, pursuing the transition between shape and number smoothly, and the readability of the algorithm is improved.

References

- [1] M. ATIYAH: *Mathematics in the 20th Century*. Bull. Lond. Math. Soc. **34**(1), 1–15, 2002. doi: 10.1112/s0024609301008566.
- [2] M. CYRUS and J. BECK: *Generalized two- and three-dimensional clipping*. Comput. Graph. **3**(1), 23–28, 1978. doi: 10.1016/0097-8493(78)90021-3.
- [3] S. W. HAWKING: *A brief history of time*. Bantam Books Inc., USA, 1 ed., 1988.
- [4] Y. J. HE: *Geometric Computing*. Higher Education Press, Beijing, China, 2013.
- [5] Y. D. LIANG and B. A. BARSKY: *A New Concept and Method for Line Clipping*. ACM Trans. Graph. **3**(1), 1–22, 1984. doi: 10.1145/357332.357333.
- [6] J. R. MILLER: *Vector geometry for computer graphics*. IEEE Computer Graph. Appl. **19**(3), 66–73, 1999. doi: 10.1109/38.761552.
- [7] W. M. NEWMAN and R. F. SPROULL: *Principles of Interactive Computer Graphics*. McGraw-Hill Inc., New York, NY, USA, 2 ed., 1979.
- [8] G. PICCININI: *Computationalism in the Philosophy of Mind*. Philosophy Compass **4**(3), 515–532, 2009. doi: 10.1111/j.1747-9991.2009.00215.x.
- [9] H. YU, Y. HE, and W. ZHANG: *A New Approach to Analyzing Interactions of Two Objects in Space Based on a Specially-Tailored Local Coordinate System*. IEEE Access **9**, 60258–60264, 2021. doi: 10.1109/access.2021.3074509.
- [10] Y. ZENG and I. HORVÁTH: *Fundamentals of next generation CAD/E systems*. Comput.-Aided Des. **44**(10), 875–878, 2012. doi: 10.1016/j.cad.2012.05.005.

Internet Sources

- [11] C. ERICSON: *Triangle-triangle tests, plus the art of benchmarking*, 2007. <http://realtimecollisiondetection.net/blog/?p=29>.

Received April 4, 2022; final form June 2, 2022.