SCIENCE CHINA
Physics, Mechanics & Astronomy

• **Research Paper** •

September 2010 Vol.53 No.9: 1610–1618
doi: 10.1007/s11433-010-4062-6

Cluster identification and characterization of physical fields

ZHANG GuangCai[*], XU AiGuo, LU Guo & MO ZeYao

National Key Laboratory of Computational Physics, Institute of Applied Physics and Computational Mathematics, Beijing 100088, China

Received November 15, 2009; accepted March 1, 2010; published online July 20, 2010

The description of complex configuration is a difficult issue. We present a powerful technique for cluster identification and characterization. The scheme is designed to treat and analyze the experimental and/or simulation data from various methods. The main steps are as follows. We first divide the space using face or volume elements from discrete points. Then, we combine the elements with the same and/or similar properties to construct clusters with special physical characterizations. In the algorithm, we adopt an administrative structure of a hierarchy-tree for spatial bodies such as points, lines, faces, blocks, and clusters. Two fast search algorithms with the complexity ln*N* are generated. The establishment of the hierarchy-tree and the fast searching of spatial bodies are general, which are independent of spatial dimensions. Therefore, it is easy to extend the method to other fields. As a verification and validation, we applied this method and analyzed some two-dimensional and three-dimensional random data.

spatial hierarchical tree, fast search, complex configuration, dynamic physical fields, cluster identification

PACS: 05.90.+m, 07.05.Kf, 07.05.Rm

Complex configuration and dynamic physical fields are ubiquitous in weapon-physics, astrophysics, plasma-physics, and material-physics. Those structures and their evolutions are characterizing properties of the corresponding physical systems. For example, the interface instability set significant constraints on the design of an inertial confinement fusion (ICF) device [1], shock waves and jet-flows in high energy physics are common phenomena [2], distributions of clouds and nebulae are very concerned issues of astrophysics [3–5], clusters and filaments occur in the interaction of high-power lasers and plasmas [6], and structures of dislocation bands determine the material softening in plastic deformation of metals [7]. These structures are also keys to understanding the multi-scale physical processes. Laws on small-scales determine the growth, the change and the interactions of stable structures on larger-scales. Description of the evolution of stable structures provides a constitutive relation for larger-scale modeling. Because of the lack of periodicity, symmetry, spatial uniformity or pronounced correlation, the identification and characterization of these structures have been challenging for years.

Existing methods for analyzing complex configurations and dynamic fields include the linear analysis of small perturbations of the background uniform field, characteristic analysis of simple spatial distributions of physical fields, etc. These methods are lacking in a quantitative description of characteristics of the physical domain. For example, the size, the shape, the topology, the circulation and the integral of related physical quantities. Therefore, it is difficult to trace the evolution of the characteristic region or the background. For example, the laws of growth and decline, or the exchange between them.

The difficulties in characteristic analysis are twofold. The first is how to define the characteristic region. The second is how to describe it. The former involves the control equations of the physical system. The latter is related to recovering the geometric structure from discrete points. In recent years, cluster analysis techniques [8,9] in data mining have found extensive applications in identification and the

*Corresponding author (email: zhang_guangcai@iapcm.ac.cn) testing of laws of targets. They mainly concern the schemes for data classification. Physicists are concerned more about the nature underlying these structures. Recovering characteristic domains can be attributed to the construction of spatial geometry. The key point is how to connect the related discrete points. The Delaunay grid [10,11] has an excellent spatial neighbor relationship. In this work, we use the Delaunay triangle or tetrahedron as the fundamental geometrical element.

The designs of the data structure and algorithm for fast searching are core issues in the field of computer software engineering [12,13]. Using a tree structure to manage spatial discrete data has obvious advantages in memory usage and fast searching. In the fields of celestial evolution and galaxy formation, a space hierarchy tree (SHT) is widely used to manage the distribution of space particles [14,15] so that the forces on particles and mass distribution of galaxies can be quickly calculated.

In this paper, we use the spatial hierarchy tree to manage objects in the *n*-dimensional space. Two general adaptive fast searching algorithms are presented. As applications, Delaunay division and cluster structure construction in two- and three-dimensional spaces are performed.

1 SHT management structure

SHT has been successfully applied to the management and indexing of spatial data points [16], but has not yet been applied to more complex spatial objects, such as lines, surfaces, bodies, clusters, etc. In this paper, we use the SHT to manage objects with spatial location, shape and size. The basic idea is as follows. For a system in the *n*-dimensional space, we design an *n*-dimensional cube to contain the system; and then divide this cube in each dimension into two parts to form 2^n sub-cubes; only retain the cubes with objects inside; continue to decompose each cube until the required resolution is reached; put the objects (points, lines, surfaces, bodies) into the appropriate cube according to their locations and sizes; existing cubes are connected together, according to their belonging relationships, to form a ‘spatial hierarchical tree’. Each cube is named a ‘branch’. Its child-cubes are named ‘sub-branches’ and its parent-cube is called ‘trunk’. The largest cube is named the ‘root’. Figure 1 (Figure 2) is a schematic for the SHT management structure of two-dimensional (three-dimensional) discrete points. Due to the uncertainty of the number of ‘sub- branches’ in a ‘branch’, ‘branches’ sharing the same ‘trunk’ are grouped as a linked list; Similarly, ‘objects’ belonging to the same ‘branch’ are also linked as a list.

In practical applications, the number of spatial objects may be variable. Therefore, the SHT is constructed dynamically. For the establishment of a ‘tree’ from an object, the typical procedure consists of two steps: (i) Get the known the minimum resolution, i.e. the smallest edge length

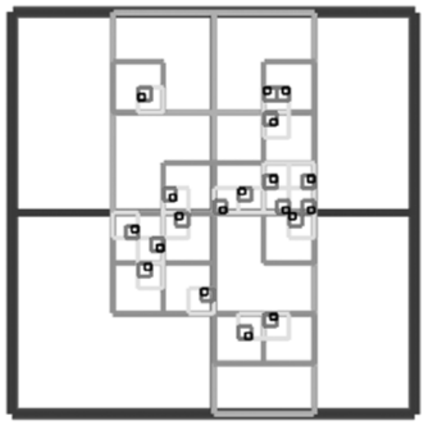
Figure 1 Management region of SHT of two-dimensional discrete points.

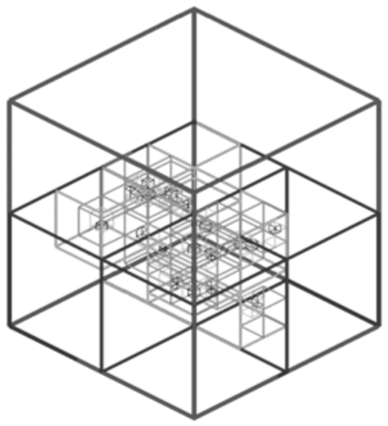
Figure 2 Management region of SHT of three-dimensional discrete points.

of cubes, σ; (ii) Use the center of an object as the geometrical center of the cube. Check whether or not the cube can contain the object. If yes, the cube is a proper ‘branch’, then put the object into this branch; if not, the cubic length will continue to double until the object can be contained, then create a ‘branch’ with the object placed in. Up to now, we have just established a ‘tree’ with only one ‘branch’ which contains one ‘object’. Figure 3 shows the construction process of the original tree, where a triangular object in two-dimensional space is used as an example.

For convenience, we use ‘A’ to represent the already existing objects (possibly more than 1) on the tree. The algorithm for adding a new object B to the tree is as follows: (i) Establish a new root. Check whether or not the object B can be contained by the old root. If not, then create a new root: calculate the quadrant where the center of object B is located. Set the vertex of the old root which is located in this quadrant as the center of the new root. In this way, the new root is the trunk of the old one. Then, the old root becomes a sub-branch of the new one. Continue this process until the new roots can contain the object B. (ii) Placement of object B. Start from the new root. Compute the quadrant where the

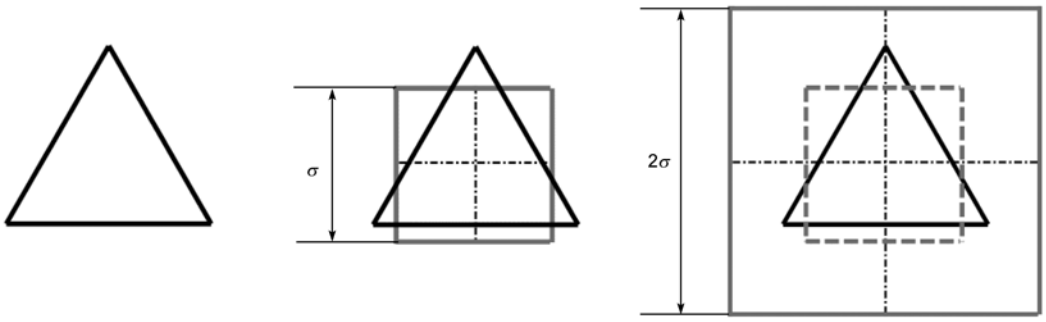

Figure 3 Schematic for the construction of the original tree.

center of object B is located, with respect to the current branch. Check whether or not the sub-branch of this quadrant contains object B. If not, the object B is placed in current branch; if yes, take the sub-branch as the current branch. If the sub-branch does not exist, create it. Continue this process until a sub-branch is found or created which can contain object B. Figure 4 is the schematic for the addition of a triangle to an existing tree.

The algorithm for removing an object from the tree is as follows: According to the link, pick up the branch containing the object. Remove the object from the object-list corresponding to this branch. When a branch no longer contains objects and sub-branches, remove it. Enter its trunk, continue this process until all the useless branches are eliminated.

In the dynamical algorithm of the SHT, except for adding a sub-branch or trunk of a branch, other operations have nothing to do with space dimension. The computer memory required by the SHT is approximately equal to $kN\ln N$, where N is the number of objects. It is independent of the spatial dimension. When the spatial dimension is higher or spatial objects have a scattered distribution, the SHT can save a large quantity of memory compared with the background grid method. In addition, because SHT is dynamically constructed the size of the system can dynamically increase or decrease with the addition or deletion of objects. This is a second obvious advantage over the traditional background grid method.

2 Fast searching algorithms based on SHT

When constructing spatial geometry or determining neighbor relationship between objects, we need a fast search of objects satisfying certain conditions. The computational complexity of an ergodic search is N. It is not practical when dealing with a huge number of objects. For such cases, we need to develop fast searching algorithms.

By using the SHT we propose a fast searcher with computational complexity $\ln N$. The basic idea is as below: We do not search the objects directly, but rather check branches. Skip those branches without objects under consideration. Thus, searching is limited to a substantially small range. Depending on requirements of applications, we present two fast searching algorithms: conditional searching and minimum searching. The goal of conditional searching is to search for objects meeting certain conditions. For example, to find objects in a given area. The goal of minimum searching is to search for an object whose function value is minimum. For example, to find the nearest object to a fixed point.

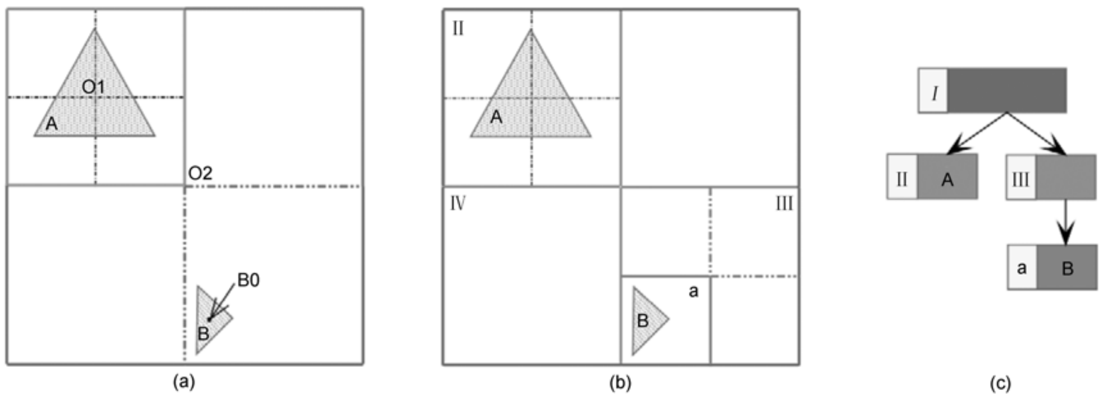

Figure 4 Schematic for adding objects to the tree. (a) Generation of a new root; (b) placing an object; (c) SHT corresponding to (b).

2.1 Conditional search

The idea of a conditional search is thus: First check whether or not a branch contains objects meeting some condition. If not, skip the branch. For example, to search for objects in a circle, one needs to assess whether or not the region of the branch intersects with the circle. In this way, the searching is limited to the overlapped region of the branch and the circle.

Conditional searching is implemented using the stack structure. The steps are as follows: (i) Push the root into a stack A; (2) Pop out a branch b from the stack A; Check whether or not the objects in b satisfies the given conditions; Pick out the required objects; (3) Check each sub-branch of b; Push the branches satisfying the conditions into stack A; (4) Repeat (2)–(3) until the stack is empty.

Figure 5 shows the given circle and spatial division for managing planar triangles. Figure 6 is the schematic for the SHT corresponding to Figure 5, where '/' stands for the root. The process to find out objects in the given circle is shown in Figure 7.

The conditional searching is implemented by providing a *conditional function* and an *identification function*. The conditional function presents conditions which the objects should satisfy. The identification function is used to assess whether or not the region of a branch contains suitable objects. It is clear that the validity of the algorithm is assured by the identification function. The more accurate the identification is, the fewer branches need to be searched. If identification status is always true, this searching algorithm goes back to an ergodic browser.

2.2 Minimum search

For convenience of description, we define a few concepts. (i) *Range of a branch*: It means the range of the given function for objects in this branch. (ii) *B-R-branch*: It is a the new branch data structure composed of the branch itself and

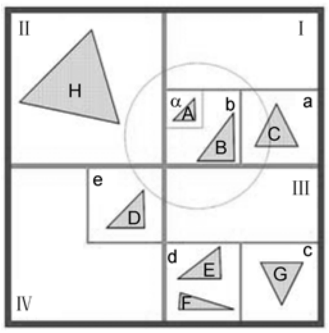

Figure 5 Distribution of planar triangles and the corresponding spatial division.

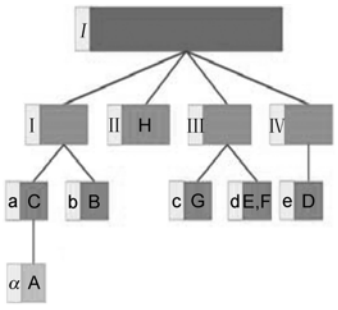

Figure 6 SHT corresponding to Figure 5.

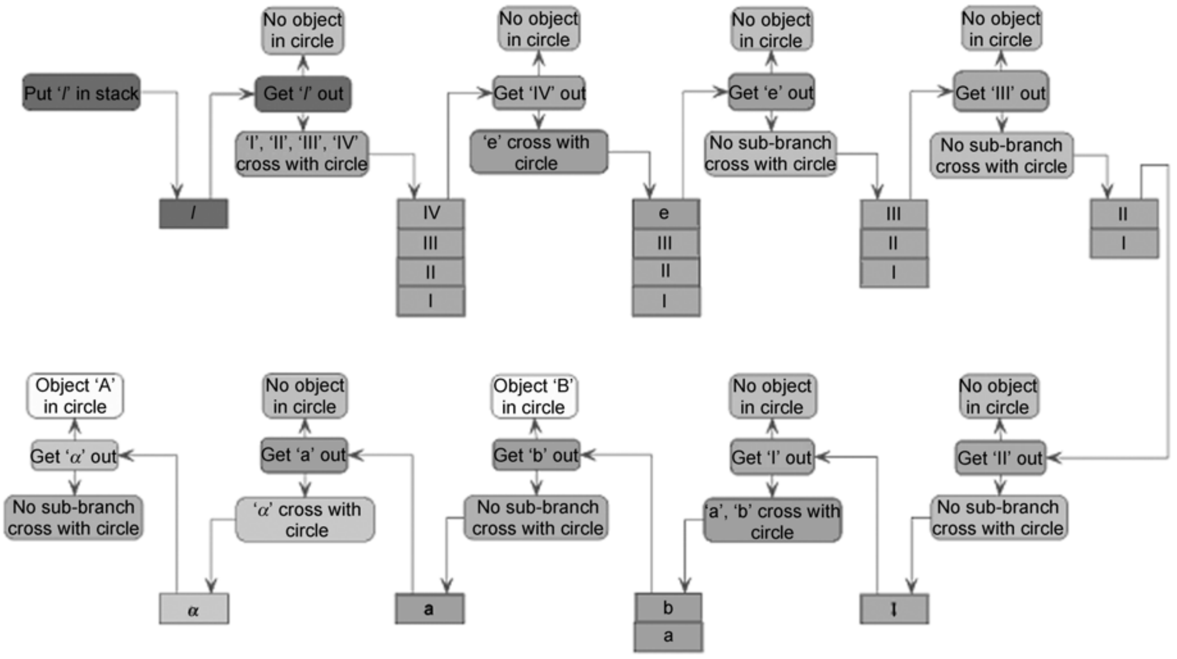

Figure 7 Flowchart for the fast search of objects in a circular area.

range of this branch. (iii) *Candidate B-R-branch*: It is the B-R-branch which may be checked in the following procedure. It may contain objects whose functional values are minimum. In the minimum searching procedure, we must keep enough candidate B-R-branches. Some of them may be dynamically added or removed according to the need. In order to accelerate the searching speed, the candidate B-R-branches should be linked as a list. According to the above definition, each B-R-branch has a range. So, each B-R-branch has a lower limit to its range. The B-R-branches in the list are arranged in such a way that their lower limits subsequently increase. Obviously, the B-R-branch with the smallest lower limit is placed at the head of the list. In addition, candidate objects and candidate values should be used to store the current objects with minimum values and the values themselves.

The idea of minimum searching is thus: By comparing the ranges of different branches, some branches can be excluded from the searching. The minimum searching algorithm is as follows: (i) The root and its range are combined as a B-R-branch; Add the B-R-branch to a candidate list named L; The candidate value V is set as positive infinity; The candidate object is set as null. (ii) Pick out a B-R-branch, for example, B, from candidate list L; Check the values of its objects. If the value of an object O is smaller than V, then, replace V with this value; in the mean time, set object O as a candidate object. Remove the B-R-branches whose lower limit values are greater than V from the list L. (iii) Construct a B-R-branch Z for each sub-branch of B. If the minimum value of Z is larger than V, cancel Z; If the maximum value of Z is smaller than the minimum value of the B-R-branch C in L, then all the B-R-branches behind C are removed from L; If the minimum value of Z is larger than the maximum value of a B-R-branch in L, cancel Z; Otherwise, insert Z into list L according to its lower limit. (4) Repeat steps (2) and (3) until the candidate list L is empty. The final candidate object is the required one.

As an example of applications of the proposed minimum search algorithm, we consider a case to find the nearest point to a fixed one from points in a plane. Figure 8 shows the distribution of planar points and spatial division. Figure 9 is the corresponding SHT. Suppose point A in Figure 8 is the given fixed point. To seek the nearest point to it, the range of the branch is calculated by a sphere evaluation method. Figure 10 shows the flow-chat.

We can perform various minimum searches by providing a different *value-finding function* and *range-evaluation function*. The value-finding function computes the value of an object. The range-evaluation function assesses the range of a branch. The efficiency of the minimum searching algorithm depends on the range-valuation function. The smaller the range given by the range-evaluation function, the faster the searching procedure. The worst range-evaluation function gives a range from $-\infty$ to $+\infty$. In such a case, the searching algorithm goes back to the ergodic browser. In the case with a large quantity of objects, one should use a good range-evaluation function to reduce the number of objects to be searched. However, a good range-evaluation generally needs a large quantity of computations, which also decreases the global efficiency. We should find a balance between the two sides. Since the computation for sphere regions is more efficient than for cube ones, in complex minimum searching algorithms, circumspheres of a cube are extensively used to evaluate the range of a branch.

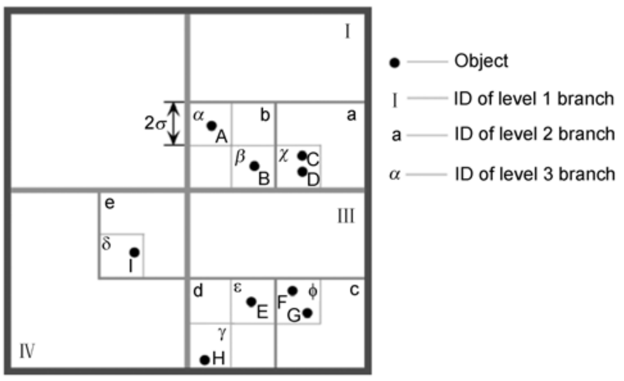

Figure 8 Planar point distribution and corresponding spatial division.

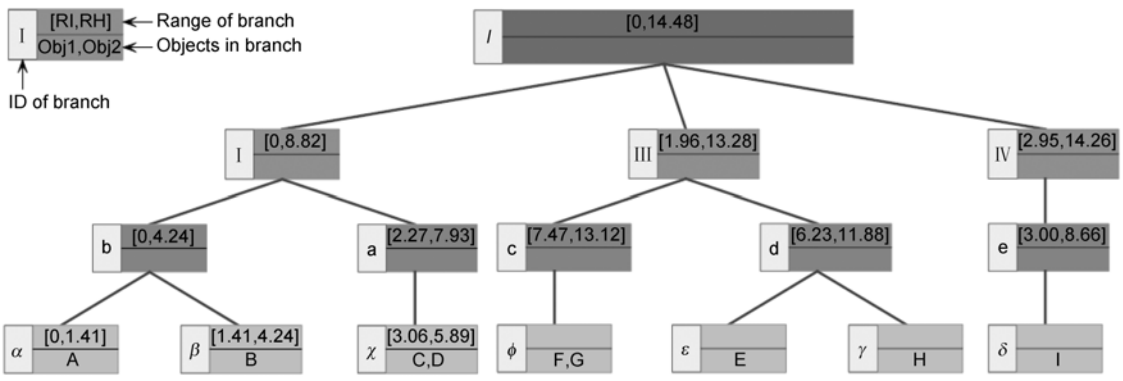

Figure 9 SHT corresponding to Figure 8.

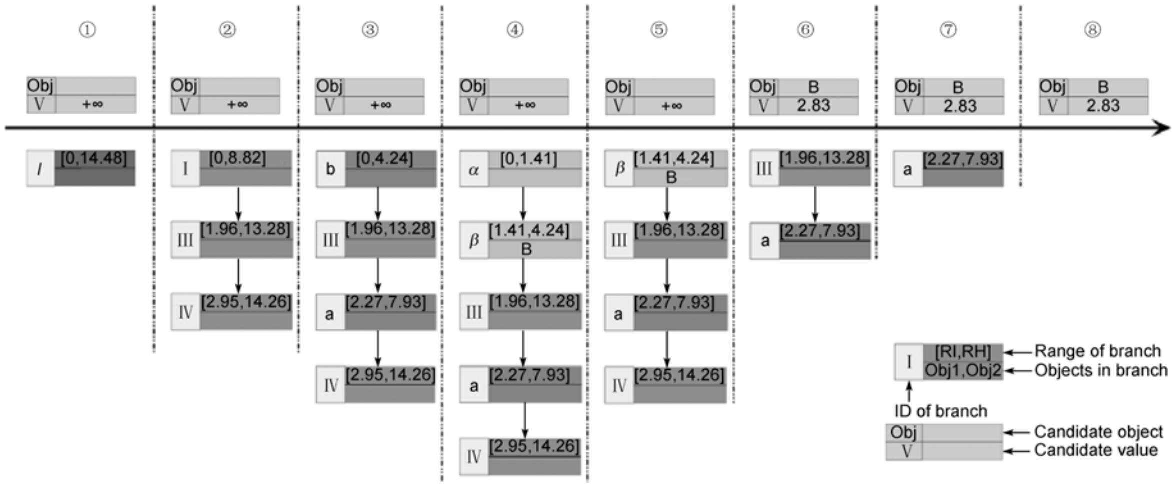

Figure 10 Schematic for the fast search of the nearest point to a given fixed one.

3 Constructing Delaunay tetrahedrons and triangles

Before constructing clusters, we divide the space using the given discrete points. We construct spatial geometrical structures by connecting given discrete points according to the Delaunay division approach. There are lots of algorithms to the construction of Delaunay tetrahedrons in three-dimensional space or Delaunay triangles in two-dimensional space. The complexities of most algorithms are involved with the searching procedures. Here, we propose an algorithm based on the SHT. The algorithm is simple and intuitive. It is convenient to extend to higher-dimensional space.

The main idea of the algorithm is as follows: when a new point is added to the formed Delaunay division structure, we should adjust the subdivision near the new point to meet the condition for Delaunay division. According to the definition of Delaunay division, if the new point is outside the circumsphere of a Delaunay simplex, this addition does not affect the Delaunay simplex. On the contrary, if it is inside the circumsphere of the Delaunay simplex, it does affect the simplex. The simplex needs to be re-divided. Specifically, all the simplexes affected by the new added point are selected to form a complex. Each face of the complex and the new point form a new simplex.

In three-dimensional space, the algorithm for the construction of the Delaunay tetrahedron from given discrete points is as follows: (i) Generate a sufficiently large tetrahedron to contain all discrete points; Record the center and radius of its circumsphere; Form an 'extended-tetrahedron' of the circumsphere; Add this extended-tetrahedron to SHT with the name T (extended-tetrahedron SHT); (2) Pick out a discrete point P; Search in T for the extended-tetrahedra whose circumspheres contain P; Remove these extended-tetrahedra from T; Put the removed tetrahedra together to form a set named Q; (3) Add every surface of each tetrahedron in Q to SHT S which is a tree for the external triangular interfaces. Remove the surfaces that appears twice because they are interfaces; Triangles in S constitute the external interface of Q; (4) Pick out each face of S, together with point P, to construct a new tetrahedron; Record the center and radius; Add the newly formed extended-tetrahedra to T; (5) Repeats steps (2) to (4) until all points are used out. The set of tetrahedra in T is just the required Delaunay division structure.

The algorithm includes two searches: one is for the circumsphere that contains a given point, the other is for the external interface of Q. They both belong to the conditional search. When we reduce to the two-dimensional case, the algorithm stays the same. We need only replace the tetrahedron with a triangle and replace the triangular face with a line. Figure 11 shows the procedure of adjusting the space division due to the addition of a two-dimensional discrete point. Figure 12 shows the Delaunay triangle division of 20000 randomly distributed two-dimensional points. Figure 13 shows the Delaunay tetrahedron division of 20000 randomly distributed points in a three-dimensional sphere.

The algorithm can be easily extended to n-dimensional space. We need only replace the tetrahedron with an n-simplex and replace the triangle with an $(n-1)$-simplex. The circumsphere of the n-simplex constructed from n+1 points, $\{\boldsymbol{r}_1, \boldsymbol{r}_2, \cdots, \boldsymbol{r}_{n+1}\}$, in n-dimensional space is used in the algorithm. The formula to calculate the center of the circumsphere of the n-simplex is $c_i = A_i / B$, where

$$A_i = \begin{vmatrix} 1 & -2\boldsymbol{r}_1 & \boldsymbol{r}_1^2 \\ 1 & -2\boldsymbol{r}_2 & \boldsymbol{r}_2^2 \\ \cdots & \cdots & \cdots \\ 1 & -2\boldsymbol{r}_{n+1} & \boldsymbol{r}_{n+1}^2 \\ 0 & \boldsymbol{e}_i & 0 \end{vmatrix}, \quad B = \begin{vmatrix} 1 & -2\boldsymbol{r}_1 & \boldsymbol{r}_1^2 \\ 1 & -2\boldsymbol{r}_2 & \boldsymbol{r}_2^2 \\ \cdots & \cdots & \cdots \\ 1 & -2\boldsymbol{r}_{n+1} & \boldsymbol{r}_{n+1}^2 \\ 0 & \boldsymbol{0} & 1 \end{vmatrix},$$

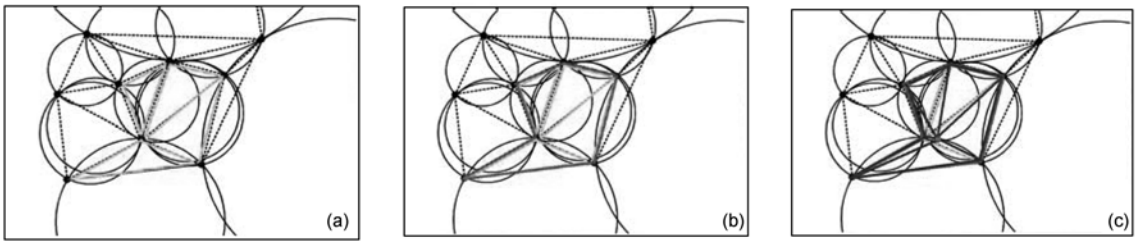

Figure 11 Three steps to add a new point to a two-dimensional Delaunay division. (a) Finding the triangles whose circumcircle contains the newly added point p; (b) removing the internal lines of these triangles, retaining the external ones; (c) connecting each left line with point p to form new triangles.

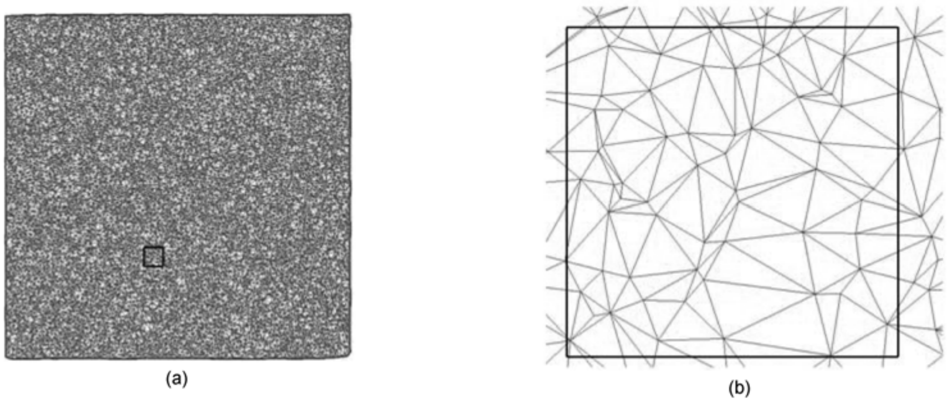

Figure 12 Delaunay division constructed from randomly distributed discrete points in a two-dimensional square area [0,4]×[0,4]. (b) is the enlarged picture of the portion in the small black rectangle in (a).

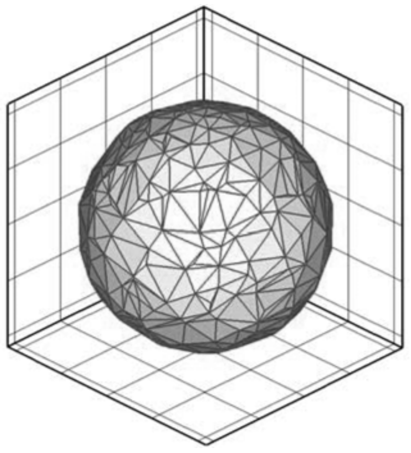

Figure 13 Delaunay division constructed from randomly distributed discrete points in a three-dimensional spherical region.

$\boldsymbol{e}_i$ is the unit vector of the i-th direction. The radius of the circumsphere is $\sqrt{(\boldsymbol{c}-\boldsymbol{r}_1)^2}$.

4 Cluster construction and analysis method

For the discrete points in space, there is no strict cluster structure. If the discrete points are considered as objects, such as a molecular ball, lattice or grid, then, the objects can be connected to form clusters. The average size of these assumed objects is the resolution of clusters to be constructed with discrete points. The construction of clusters is very simple. A cluster is formed by connecting all points whose distance in between is less than the resolution length.

After the construction of the Delaunay division for given set of discrete points, remove the lines whose lengths are greater than the resolution length. The remaining spatial structure may have various dimensions. According to connectivity, the structures that are not connected to each other can be decomposed into different clusters. Each cluster may also have structures with various dimensions. For example, a structure consisted of two triangles with a common side, or a structure formed by a triangle and a tetrahedron, etc. In physical problems, the structures with high-dimensional measures play a major role in describing the system. Generally, we need only need to analyze clusters with the maximum dimensions.

The cluster construction algorithm consists of three parts. Preparation part: (i) Construct Delaunay tetrahedra from given discrete points. The corresponding SHT is notated as t. (ii) Remove the tetrahedrons whose length is greater than the given resolution from t. Single cluster construction part: (iii) Remove tetrahedron T from t if such a T still exists. Create a new cluster named C. Initialize the body tree C->t and face tree C->s as null. Add T to the body tree C->t. Add each of the triangle faces to a triangle tree named i. (iv) Pick

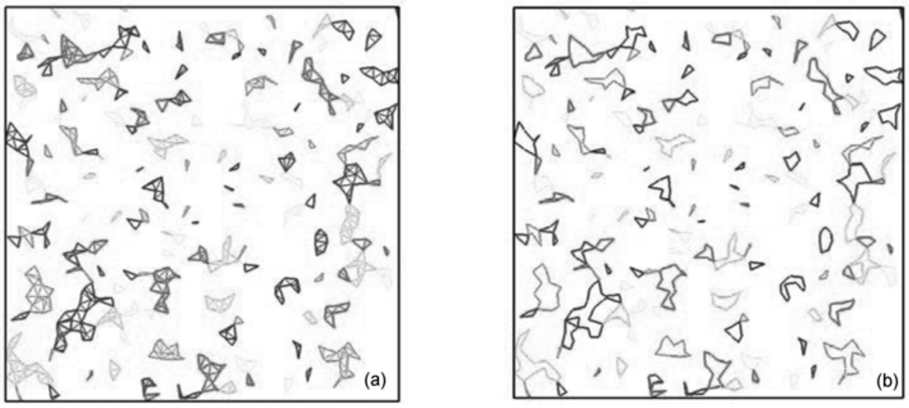

Figure 14 Cluster structure formed from 1000 random discrete points in a two-dimensional square area [0,1]×[0,1]. (a) A cluster; (b) the corresponding cluster boundary.

out a triangle face S from i. Search tetrahedron Y containing face S from t. If found, add Y to tree C->t and add all faces of Y to tree i. Two faces with opposite directions will annihilate if they meet each other during the adding procedure. If none are found, add S to the tree C->s. (v) Repeat step (iv) until the tree i becomes null. Construct all the clusters: (vi) Add the constructed cluster C to a tree for clusters named c. Repeat the process of constructing single clusters, and add the new cluster to c until t becomes null. The algorithm for adding a face S to the tree i is as follows. Search and check if a face with the opposite direction of S exists in the tree i. If it exists, remove it from the tree i. If it does not exist, add S to the tree i. Up to now, all the constructed clusters are put to the tree for clusters c. For each cluster C in the tree c, all tetrahedron elements are placed on the body tree C->t, all the surface triangles are placed on the tree for faces C->s. Figures 14 and 15 show respectively the clusters constructed with random points in two-dimensional and three-dimensional space.

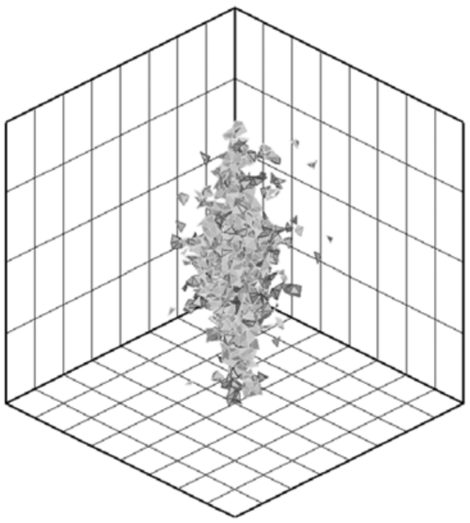

Figure 15 Cluster structure formed from 5000 three-dimensional random discrete points.

The algorithm is also applicable to n-dimensional discrete points. We need only replace the tetrahedron with an n-simplex and replace the triangular surface with an $(n-1)$-simplex. For space with a dimension higher than three, the number of neighboring points and the connectivity, as well as the number of n-simplex, grow rapidly with the dimension. So, the required memory increases quickly. The Delaunay division can be constructed partition by partition. The main skill in this algorithm is that the space is partitioned according to the main branches of SHT, and points in each partition are added sequentially. After the completion of adding all points in a partition, we need to delete the n-simplex that satisfies two conditions: (i) its external circumsphere is in the completed partition, (ii) at least one side is longer than the given resolution.

5 Conclusions and discussion

We propose a new method for managing objects and fast searching in arbitrary dimensional space. Based on this, algorithms for the constructing a Delaunay simplex and spatial clusters are presented. The applications to two- and three-dimensional discrete points validate the method and show obvious advantages.

The proposed SHT can be easily used to manage and search objects with various locations, sizes or even shapes. This management method can be widely used in many fields. As an example, in the finite-element method, due to the complexity in calculating relative positions of elements and in the search for them, only the cases with simple shapes and single-sizes are extensively studied. With the proposed SHT, it is easy to search for adjacent relationships between objects with various sizes and shapes. Therefore, the SHT can substantially simplify simulations with the movements of complex objects.

Compared with previous methods, the proposed SHT and the two fast-searching algorithms based on it are established on a more abstract framework. It has potential extensibility to various fields. Experimental data can also be treated under the same SHT after parameterization of physical quantities. Then, fast searching can be realized according to a similar algorithm. In addition, according to the needs of the user, an object can be simultaneously placed in several SHTs. This is equivalent to setting indexes of several properties. Therefore, fast searching for various properties can be easily implemented.

This work was supported by the National Natural Science Foundation of China (Grant Nos. 10702010 and 10775018) and the Science Foundations of the Laboratory of Computational Physics and China Academy of Engineering Physics (Grant Nos. 2009A0102005 and 2009B0101012).

1 Ament P. Effects of ionization gradients on Inertial-Confinement-Fusion capsule hydrodynamics stability. Phys Rev Lett, 2008, 101: 115004

2 de Vries1 P C, Hua M D, McDonald D C, et al. Scaling of rotation and momentum confinement in JET plasmas. Nucl Fusion, 2008, 48: 065006

3 Bowler B P, Waller W H, Megeath S T, et al. An infrared census of star formation in the horsehead nebula. Astron J, 2009, 137: 3685–3699

4 Hernquist L. Hierarchical *N*-body methods. Comput Phys Commun, 1988, 48: 107–115

5 Makino J. Vectorization of a treecode. J Comput Phys, 1990, 87: 148–160

6 Hidaka Y, Choi E M, Mastovsky I, et al. Observation of large arrays of plasma filaments in air breakdown by 1.5-MW 110-GHz gyrotron pulses. Phys Rev Lett, 2008, 100: 035003

7 Nogaret T, Rodney D, Fivel M, et al. Clear band formation simulated by dislocation dynamics: Role of helical turns and pile-ups. J Nucl Mater, 2008, 380: 22–29

8 Kotsiantis S B, Pintelas P E. Recent advances in clustering: A brief survey. WSEAS Trans Inform Sci Appl, 2004, 1: 73–81

9 Fan Y J, Iyigun C, Chaovalitwongse W A. Recent advances in mathematical programming for classification and cluster analysis. CRM Proc Lecture Notes, 2008, 45: 67–93

10 Chazelle B, Devillers O, Hurtado F, et al. Splitting a Delaunay triangulation in linear time. Algorithmica, 2002, 34: 39–46

11 Clarkson K L, Varadarajan K. Improved approximation algorithms for geometric set cover. Discrete Comput Geom, 2007, 37(1): 43–58

12 Black P E. Entry for data structure in Dictionary of Algorithms and Data Structures. U.S. National Institute of Standards and Technology. 15 December. 2004

13 Knuth D E. The Art of Computer Programming. Vol. 3: Sorting and Searching. Redwood City: Addison Wesley Longman Publishing Co, 1998

14 Barnes J, Hut P. A hierarchical $O(N\ln N)$ force-calculation algorithm. Nature, 1986, 324: 446–449

15 Pfalzner S, Gibbon P. Many Body Tree Methods in Physics. New York: Cambridge University Press, 1996

16 Nam B, Sussman A. A comparative study of spatial indexing techniques for multidimensional scientific dataset. In: Proceedings of the 16th International Conference on Scientific and Statistical Database Management (SSDBM'04), June 21-23, 2004. 171