

An Efficient Convex Hull Algorithm Based on Elastic Ellipsoid in Three-dimensional Space

Changyuan Xing^{1,2}

¹College of Computer Science, Chongqing University, Chongqing 400044, China

²College of Computer Engineering, Yangtze Normal University, Fuling 408100, Chongqing, China

Zhongyang Xiong

College of Computer Science, Chongqing University, Chongqing 400044, China

Xuegang Wu^{1,2}

¹College of Communication Engineering, Chongqing University, Chongqing 400044, China

²College of Computer Engineering, Yangtze Normal University, Fuling 408100, Chongqing, China

Fengkuan Xu, Wenjun Yang

College of Communication, Hulunbeier University, Hulunbeier 021000, Inner Mongolia, China

Abstract

Along with the development of computer simulation, virtual reality and pattern recognition, the massive data collection in three-dimensional (3D) space has gradually increased. To solve the issue of fast convex hull (CH) computation of 3D massive data, an efficient convex hull algorithm inspired by visual attention mechanism is proposed, in which the tetrahedron used in the traditional method is replaced by an elastic ellipsoid. As removing non-vertex points (not belong to a convex hull) as many as possible in the initial step, our algorithm can attain the 3D convex hull quickly. The experiments in several typical data sets show that the proposed algorithm can be applied to the 3D massive data, it only uses 1.78 seconds to get convex hull for ten million point under multivariate normal distribution, and it achieves a better performance than Chan's and Quickhull algorithms in terms of time consumption and memory usage when the data number of 3D point set increases.

Key words: PATTERN RECOGNITION, SHAPE ANALYSIS, COMPUTATIONAL GEOMETRY, CONVEX HULL

1. Introduction

Three-dimensional convex hull is a fundamental structure in computational geometry (see Fig. 1). It has been widely used in many fields, for example, pattern recognition [1], clustering [2], data mining [3], image processing [4], virtual reality [5], acoustic contrast [6] and outlier detection [7]. Especially, in the recent years, 3D convex hull is useful to compute accessibility map in GIS. Magnetic resonance imaging (MRI) in 3D can be outlined the possible lesion areas for doctors based on convex hull algorithm.

Since Chand and Kapur initially proposed a CH algorithm with $O(n^2)$ time, many algorithms have been proposed in three-dimensional space [8, 9, 10, 12].

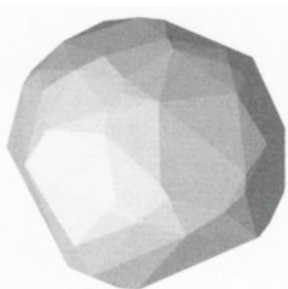


Figure 1. A three-dimensional convex hull

Graham [8] proposed an approach with $O(n \log n)$ time to compute the CH of a finite point set in a two-dimensional (2D) plane. A variant on an $O(n^2)$ sorting algorithm was given by Jarvis [9] in 3D, which built the hull in $O(nh)$ time by a process called gift-wrapping, where h is the CH vertex number. Preparata and Hong [10] proposed a recursive algorithm based on divide-and-conquer strategy with $O(n \log n)$ time complexity. Clarkson and Shor [11] introduced an incremental insertion algorithm which deals with new points in turn based on the current convex hull set. If the points in the interior of the convex hull set, these points would be discarded. Barber [12] inspired by Clarkson and Shor, and then proposed Quickhull algorithm which is considered to be fast and efficient in 3D space. The time complexity of the incremental insertion algorithm and Quickhull algorithm are $O(n \log n)$. Chazelle and Matousek [13] described a 3D convex hull algorithm based on random increment, in which the points were processed in a random order. Chan [14] proposed an output sensitive 3D convex hull algorithm, the time complexity is $\theta(n \log h)$.

Due to the emergence of parallel computer technology in the recent decades, the algorithms for parallel 3D convex hull computation were described. Most of them are based on the Parallel Random-access Machine (PRAM) model. Miller and Stout [15] proposed

a fast parallel 3D convex hull algorithm which used a hypercube, vertebrae, tree and mesh to determine the extreme points of convex hull. Amato and Preparata [16] presented a 3D convex hull algorithm based on $n^{1+\alpha}$ processor, where $\alpha \in (0,1]$, with the time complexity $O(\log n)$. Gupta and Sen [17] also proposed an output sensitive parallel algorithm, in which the problem of concurrent read-and-write conflicts was solved. The PRAM model is based on shared memory, thus the delay of memory storage is vital to synchronous operation of the processor. With the help of CUDA programming model, Cao [18] proposed gHull algorithm which used 3D voronoi map and convex hull to achieve parallelism on GPU and Stein [19] described a CudaHull algorithm based on Quickhull.

Quickhull algorithm tends to perform well in practice, which is regarded as a classical way to get convex hull in 3D [20]. The tetrahedron is initialized by computing four extreme points which are vertexes of convex hull. These four points are named by a , b , c and d separately. All points situated in tetrahedron $abcd$ will be excluded from further consideration. Residuary points will lie out facets abc , bcd , cda and dab respectively. When the farthest point e from facet abc has to be found, the points in new tetrahedron $abce$ are deleted. For each of other points lying out four facets, they does the same as above. Recursive procedure is performed to return the corresponding vertices in a given subset.

One key operation in 3D Quickhull algorithm is to check whether the points inside a tetrahedron. We notice that this operation always takes much time in the initial step, especially for the massive data. One characteristic of human visual attention mechanism [21] is that when tries to find the convex hull of a point set, human only pays attention to the points near the boundary and neglects most inner non-vertex points by an initial estimation of the boundary in the point set. Our algorithm imitates this characteristic and improves the performance of the proposed algorithm which uses an initial estimation of the boundary of the 3D point set and a maximum volume inscribed ellipsoid (MVIE) to remove most irrelevant points as a processing step.

In this paper, we extend the existing research [22] from two-dimension to three-dimension. The implementation of our approach is: firstly, some extreme points of a 3D point set will be obtained to construct an initial boxing convex polyhedron (IBCP), which is an initial estimation of the boundary of the point set; secondly, the interior points in MVIE are removed; finally, the remaining points outside IBCP are pro-

cessed recursively by Quickhull and all vertexes of convex hull will be found.

The rest of the paper is organized as follows. Section 2 explains the proposed algorithm. In Section 3, we present experimental results. Finally, we conclude the study.

2. Proposed algorithm

In human daily life, most of the information is obtained from visual system. In order to deal with the information effectively, human visual model has the ability to select and filter the information quickly from the complex scene, and pay attention to a few obvious objects, called visual attention mechanism.

For a given set of 3D points, human can easily locate the boundary of the set by visual attention mechanism, and filter out the interior points. Inspired by this mechanism, we first calculate several extreme points to estimate the initial distribution of the point

set. Then, the proposed algorithm will pay attention to IBCP and we will try to remove all points in IBCP.

2.1 Overview of the algorithm

In our algorithm, a 3D point set S , including n points, is regarded as the input. Then, the following five steps are to process S . Finally, the algorithm outputs the vertex set of convex hull.

Step 1: Initialize initial boxing convex polyhedron by computing fourteen extreme points.

Step 2: Calculate the maximum volume inscribed ellipsoid of IBCP.

Step 3: Exclude inner points in MVIE.

Step 4: Neglect inner points near the boundary of IBCP.

Step 5: Analyze the remaining points by Quickhull and attain all vertexes of convex hull.

The process flow is illustrated in Fig.2.

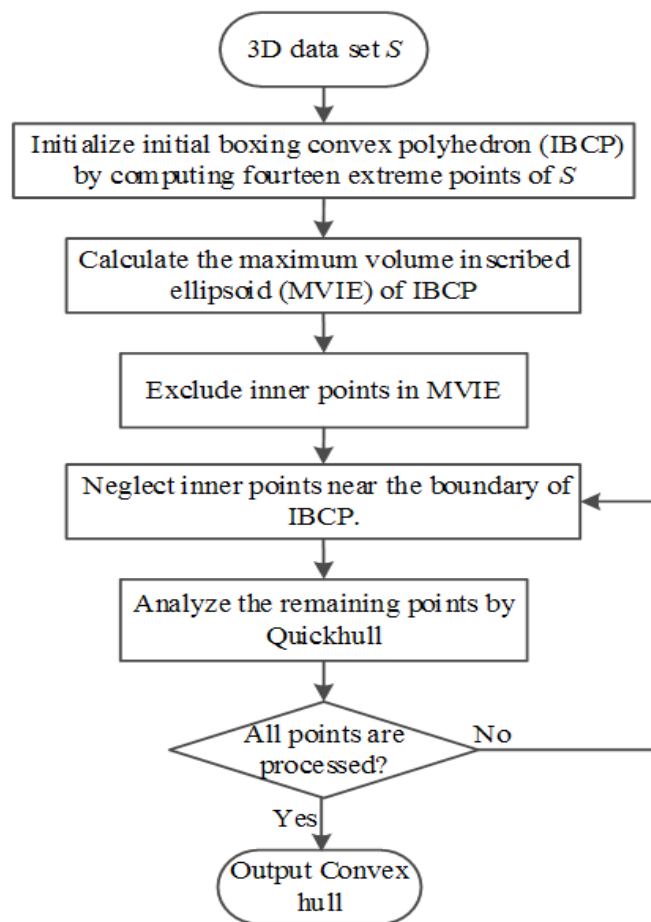


Figure 2. The processing flow of the proposed algorithm

2.2 Initializing initial boxing convex polyhedron

We compute fourteen extreme points of S as the vertexes of IBCP. These extreme points can be obtained by the following calculations:

(1) a point with maximum x -coordinates, (2) a

point with minimum x -coordinates, (3) a point with maximum y -coordinates, (4) a point with minimum y -coordinates, (5) a point with maximum z -coordinates, (6) a point with minimum z -coordinates, (7) a point that maximizes the function $f_{++}(x,y,z)=x+y+z$, (8) a point that minimizes $f_{++}(x,y,z)=x+y+z$, (9) a point

that maximizes $f_-(x,y,z)=x-y-z$, (10) a point that minimizes $f_-(x,y,z)=x-y-z$, (11) a point that maximizes $f_+(x,y,z)=x-y+z$, (12) a point that minimizes $f_+(x,y,z)=x-y+z$, (13) a point that maximizes $f_+(x,y,z)=x+y-z$, and (14) a point that minimizes $f_+(x,y,z)=x+y-z$, where x , y and z are x -coordinates, y -coordinates and z -coordinates respectively.

Let $P_{ep} = \{(x_{ep_1}, y_{ep_1}, z_{ep_1}), (x_{ep_2}, y_{ep_2}, z_{ep_2}), \dots, (x_{ep_n}, y_{ep_n}, z_{ep_n})\}$, where $ep_n \in [6, 14]$. These fourteen extreme points are the vertexes of convex hull [23]. Since there may be some duplicate points among them in some distributions, the number ep_n is usually more than or equal to six and less than or equal to fourteen. The IBCP of these extreme points are composed of some facets, each of the facet is a triangular plane. Usually, the number f_{ep} of facets is more than or equal to ep_n , e.g. in Fig.3, $ep_n = 14$, $f_{ep} = 24$.

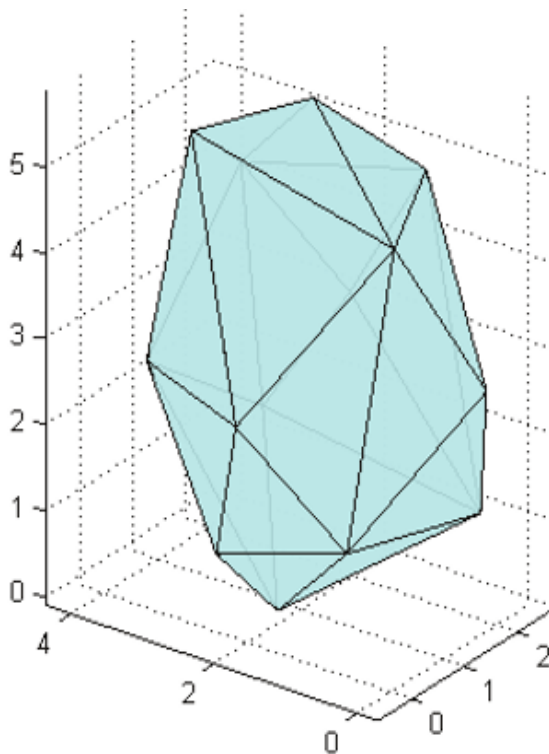


Figure 3. The initial boxing convex polyhedron (IBCP) of the 3D point set

According to statistics, more than 90% points of input point set are in IBCP. Especially, the multivariate normal distribution is over 99%. So it is the key to improve the efficiency of the algorithm by considering how to remove the interior points in IBCP quickly. We can use an inscribed ball of IBCP to remove the interior points, but IBCP is the irregular polyhedron and the volume propitiation of its inscribed ball to IBCP is not satisfied in most distributions of the data set. It means that the inscribed ball can't contain

as many as points in IBCP. Thus, we use the maximum volume inscribed ellipsoid instead of the inscribed ball to cover most of inner points of IBCP.

For different distributions in some point sets, the parameters of the ellipsoid function may be changed. This ellipsoid is called elastic ellipsoid which parameters changes with the deformation action of convex polyhedron.

Meanwhile, because most points are located around, we need to consider the centroid of a point set in our algorithm. Let $O(x_0, y_0, z_0)$ be the centroid of the point set.

2.3 Calculating the maximum volume inscribed ellipsoid

Now we consider how to attain the elastic ellipsoid (maximum volume inscribed ellipsoid) which lies inside IBCP. There are always exist a MVIE of IBCP [24]. IBCP is regarded as a polyhedron described by a set of linear inequalities

$$IBCP = \{x \mid a_i^T x \leq b_i, \quad i = 1, \dots, m\} \quad (1)$$

where $m = f_{ep}$, and the parameters a_i and b_i can be obtained from the facets of IBCP.

We parameterize the ellipsoid as the image of the unit ball under an affine transformation, i.e., as

$$E(B, d) = \{Bu + d \mid \|u\|_2 \leq 1\} \quad (2)$$

where $d \in R^n$ is the center of ellipsoid. Let L_n be a $n \times n$ symmetric matrix, L_n^+ be a $n \times n$ positive definite symmetric matrix, $B \in L_n^+$ and $B = B^T > 0$. The volume of $E(B, d)$ is proportional to $\det B$. We can find the maximum volume inscribed ellipsoid inside IBCP by solving the following convex optimization problem

$$\begin{aligned} & \text{maximize} \quad \log \det B \\ & \text{subject to} \quad B = B^T > 0, \\ & \quad \quad \quad E \subseteq IBCP, \end{aligned} \quad (3)$$

where $\log \det B$ is the maximum likelihood estimate of the determinant of matrix B .

We express the constraint in a more convenient form

$$\begin{aligned} E \subseteq IBCP & \Leftrightarrow \sup_{x \in E} a_i^T x \leq b_i, \quad i = 1, \dots \\ & \Leftrightarrow \sup_{\|u\|_2 \leq 1} a_i^T Bu + a_i^T d \leq b_i, \\ & \Leftrightarrow \|Ba_i\|_2 + a_i^T d \leq b_i, \quad i = \end{aligned} \quad (4)$$

The formula (3) is reformulated as the following form

$$\begin{aligned} & \text{minimize} \quad \log \det B^{-1} \\ & \text{subject to} \quad \|Ba_i\|_2 + a_i^T d \leq b_i, \\ & \quad \quad \quad B = B^T > 0, \end{aligned} \quad (5)$$

where the variables B and d are unknown. According to the feature of the points in 3D space under different distributions, we consider the centroid of a point set in our algorithm and suppose the center of $E(B, d)$ is the centroid of the point set, and let $d = O$. The parameter B is only unknown.

Let B is a 3×3 positive definite symmetric matrix. Then, we know that $B * B$ is also positive definite symmetric matrix. L is a lower triangular matrix based on Cholesky decomposition, and $B * B = L * L^T$. Thus, we further simplify the formula (5) as minimize $-\log \det L$

$$\text{subject to } \|L^T a_i\|_2 \leq b_i - a_i^T d, \quad i = 1, \dots, m. \quad (6)$$

The specific form of L is

$$L = \begin{bmatrix} l_0 & 0 & 0 \\ l_1 & l_3 & 0 \\ l_2 & l_4 & l_5 \end{bmatrix}, \quad (7)$$

where l_j is unknown, $j = 0, \dots, 5$.

Then, $\det L = l_0 * l_3 * l_5$. The vector a_i donates the linear inequalities coefficients from i th facet in IBCP, and the coefficients a_i is the following form

$$a_i = \begin{bmatrix} a_{i0} \\ a_{i1} \\ a_{i2} \end{bmatrix}, \quad i = 1, \dots, m \quad (8)$$

and

$$L^T a_i = \begin{bmatrix} a_{i0}l_0 + a_{i1}l_1 + a_{i2}l_2 \\ a_{i1}l_3 + a_{i2}l_4 \\ a_{i2}l_5 \end{bmatrix}, \quad i = 1, \dots, m. \quad (9)$$

Let K_i be a 3×6 matrix

$$K_i = \begin{bmatrix} a_{i0} & a_{i1} & a_{i2} & 0 & 0 & 0 \\ 0 & 0 & 0 & a_{i1} & a_{i2} & 0 \\ 0 & 0 & 0 & 0 & 0 & a_{i2} \end{bmatrix}, \quad i = 1, \dots, \quad (10)$$

and $x = [l_0 \ l_1 \ l_2 \ l_3 \ l_4 \ l_5]^T$, then the constraint in the formula (6) can be transformed to the following form

$$\begin{aligned} \|L^T a_i\|_2 \leq b_i - a_i^T d &\Leftrightarrow (K_i x)^T * (K_i x) - t_i^2 \leq 0, \quad i = 1, \dots, m \\ &\Leftrightarrow g(K_i x) - t_i^2 \leq 0, \quad i = 1, \dots, m \end{aligned} \quad (11)$$

where $t_i = b_i - a_i^T d$, $g(u) = u^T * u$.

The formula (6) can be rewritten as minimize $-\log x(0) - \log x(3) - \log x(5)$

$$\text{subject to } g(K_i x) - t_i^2 \leq 0, \quad i = 1, \dots, m. \quad (12)$$

From the formula (12), the objective function and the constraint function are twice differentiable. By transforming the inequality constraints in the formula

(12) to equality constraints (linear programming criteria), then we can solve the optimization problem by Newton method.

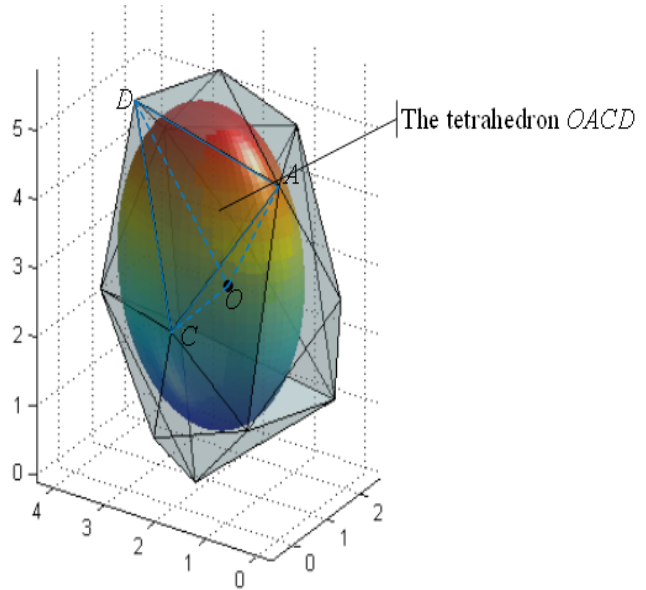


Figure 4. The maximum volume inscribed ellipsoid (MVIE) of IBCP

2.4 Excluding non-vertex points

In the above subsection, we have obtained maximum volume inscribed ellipsoid in IBCP (see Fig. 4) and supposed that the parameter, the center of the ellipsoid, was the centroid of the point set. Other three parameters of the ellipsoid are a, b and c , which donate the equatorial radius along x axis, the equatorial radius along y and the polar radius along z respectively. These three parameters are all positive real numbers, which determine the shape of the ellipsoid, and they can be computed by eigenvalues of B .

$$[a, b, c] = \text{eig}(B) = \text{sqrt}(\text{eig}(L * L^T)) \quad (13)$$

Then, the four real parameters of the ellipsoid are determined. To check whether a point $p(x_p, y_p, z_p)$ of point set S is or not in the ellipsoid, it depends on the following formula (14). If D_p is less than or equal to zero, the point p is inside or on the ellipsoid, in other cases, p is outside the ellipsoid. Compared with other 3D convex hull algorithms, the algorithm, using inscribed ellipsoid to determine the position of the point, is simple and time-saving.

$$D_p = \frac{(x-x_0)^2}{a^2} + \frac{(y-y_0)^2}{b^2} + \frac{(z-z_0)^2}{c^2} - 1 \quad (14)$$

$$\begin{cases} D_p \leq 0, & p \text{ inside or on the ellipsoid,} \\ D_p > 0, & p \text{ outside the ellipsoid.} \end{cases} \quad (15)$$

Although some points which located outside the ellipsoid, they are inside IBCP. And, these points are

not also vertexes of the convex hull. These non-vertex points will locate in the tetrahedrons which are formed by the facets and the centroid, see Fig.4.

In Fig.4, we give an example of a tetrahedron $OACD$. The distribution case of inner points in this tetrahedron is shown in Fig.5. After the removal of

the majority of the points in the ellipsoid, the rest points (labeled with solid points) in $OACD$ are near vertices A , C and D . The hollow-labeled points are outside the tetrahedron $OACD$, and they will be processed by Quickhull algorithm.

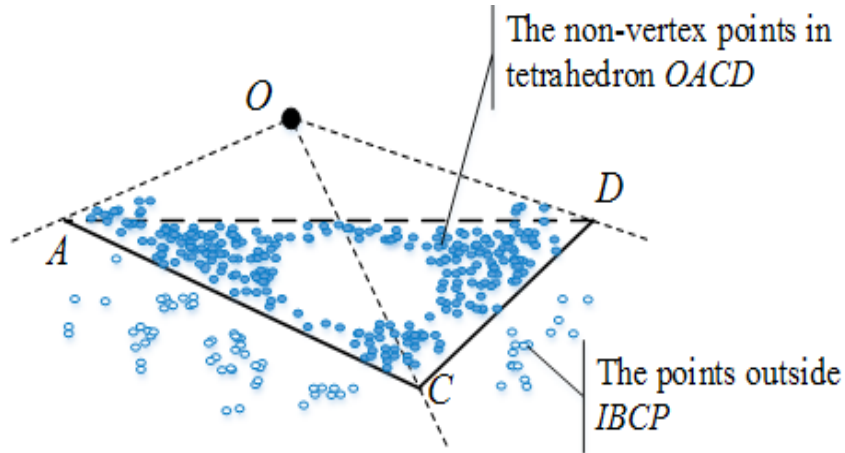


Figure 5. The distribution of non-vertex points in tetrahedron $OACD$

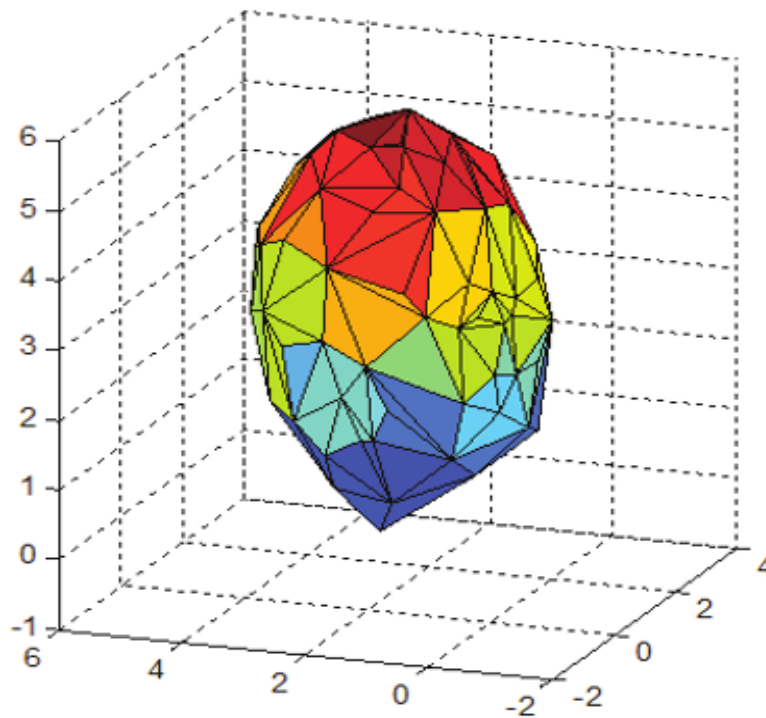


Figure 6. The final 3D convex hull when the algorithm terminated

In Fig.5, the following steps are used to judge that a solid point p located on or inside $OACD$.

(1) Let $f_i(x, y, z) = A_i \cdot x + B_i \cdot y + C_i \cdot z + D_i$ be four surface equations of the tetrahedron $OACD$, and $i = 1, \dots, 4$.

(2) The point coordinates of p will be substituted into the four surface equations $f_i(x, y, z)$. If the four values are with the same sign, the point is in the tet-

rahedron. And, if one of the four values is zero, the point is on the facet of the tetrahedron.

After excluding the points in tetrahedron $OACD$, the algorithm will process other $(f_{ep} - 1)$ tetrahedrons until all non-vertex points in IBCP are discarded.

In Step 5, the remaining points of S will be analyzed by Quickhull algorithm, then all the vertexes of the convex hull can be found (see Fig.6).

Algorithm 1 The proposed algorithm

Input: $S = \{(x_1, y_1, z_1), (x_2, y_2, z_2), \dots, (x_n, y_n, z_n)\}$, $n \in \mathbb{Z}$.

Output: $CH = \{(x_{CH_1}, y_{CH_1}, z_{CH_1}), (x_{CH_2}, y_{CH_2}, z_{CH_2}), \dots, (x_{CH_n}, y_{CH_n}, z_{CH_n})\}$, $CH_n \in \mathbb{Z}$.

1. Compute fourteen extreme point set P_{ep} of S as the vertexes of IBCP, and get the position of $O(x_0, y_0, z_0)$.
2. Calculate the maximum volume inscribed ellipsoid, transform the formula (3) to (12), and obtain the eigenvalues of B .
3. Reference formula (14) and (15), if the point is inside the ellipsoid, it will be discarded from S .
4. Exclude the remaining points which inside or on the tetrahedron $OACD$ (see Fig.5), process other $(f_{ep} - 1)$ tetrahedrons.
5. Call Quickhull algorithm to process the residual points of the point set until all points in S are judged, then output the result set of CH .

3. Experimental results

In this section, we present various experiments to evaluate the efficiency of the proposed algorithm. We will first check the ability of the proposed algorithm in excluding the proportion of points on the data sets. Then, our algorithm will be used to construct 3D convex hull in different data sets. Next, we compare time consumption and memory usage with Chan's and Quickhull algorithms. Finally, we analyze the time complexity of the proposed method.

3.1 Experiment Setup

We use data sets from six distributions generated by functions in Python program to compare the proposed algorithm with other algorithms in experiments. They are defined as

$$\mu = [1 \ 2 \ 3],$$

$$\Sigma = \begin{bmatrix} 0.1 & 0 & 0 \\ 0 & 0.2 & 0 \\ 0 & 0 & 0.3 \end{bmatrix},$$

$$S1 = \text{mvnrnd}(\mu, \Sigma, \text{num}),$$

$$S2 = \text{unifrnd}(0, 1, \text{num}, 3),$$

$$S3 = \text{exprnd}(5, \text{num}, 3),$$

$$S4 = \text{evrnd}(1, 2, \text{num}, 3),$$

$$S5 = \text{lognrnd}(3, 2, \text{num}, 3),$$

$$S6 = \text{johnsrnd}([-1.7 \ -0.5 \ 0.5 \ 1.7], \text{num}, 3).$$

Here, num is the amount of points in the point set S . Function mvnrnd returns a matrix of random vectors chosen from the multivariate normal distribution with mean μ and covariance Σ . The functions unifrnd , exprnd , evrnd , lognrnd , and johnsrnd return a matrix of random vectors generated from the uniform distribution, the exponential, the extreme value, the lognormal, and the distribution in the Johnson system respectively (see Fig. 7).

We compare the proposed algorithm with the following algorithms: Chan's and Quickhull. The process is implemented with Python 2.7.6 and all experiments are performed on a Desktop running Windows

XP with Intel Core i3 2.2 GHz and 1 GB memory.

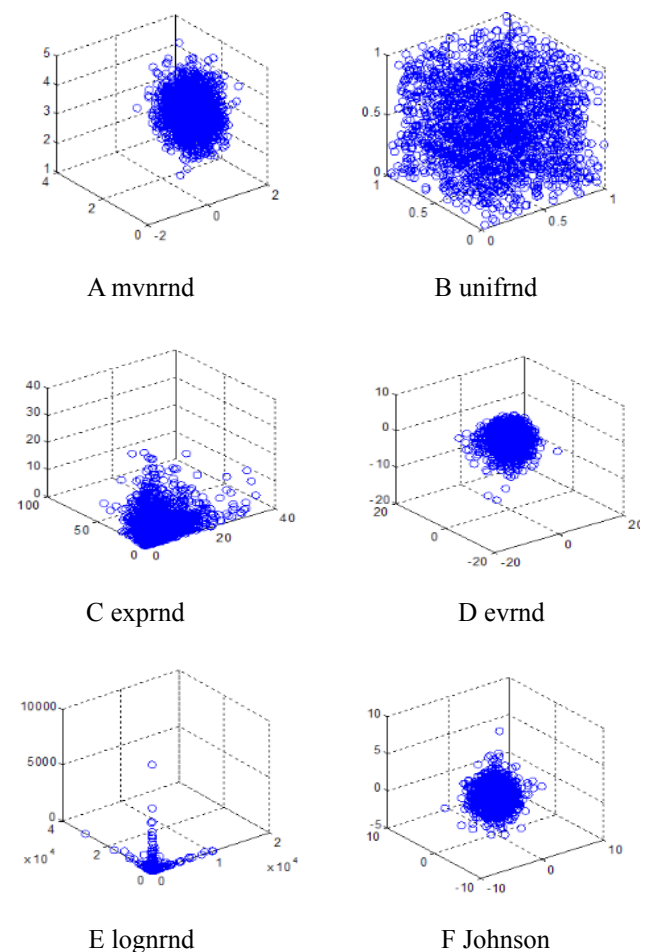


Figure 7. Six types of data sets in three-dimensional space

3.2 The proportion of points excluded in six distributions

We conduct a series of experiments to test proportions of points excluded by our algorithm. It is shown that more than 70% points are excluded by MVIE for all six data sets under different distributions, excepting the exprnd distribution and the unifrnd in Table 1. For the mvnrnd distribution and the johnsrnd , over 97% points are excluded. However, the proposed al-

gorithm is not work well in *exprnd* distribution because most of the points locate at boundary of IBCP and the inscribed ellipsoid volume is not large enough.

In each distribution, 200,000 three-dimensional points are randomly generated, we compare the differences of excluded proportions between elastic ellipsoid and inscribed ball in IBCP. Elastic ellipsoid

achieves 97.88% on *mvnrnd*, outperforming inscribed ball by a margin of 14.61%. In *lognrnd* distribution, the excluded rate with elastic ellipsoid is 79.29%, more than 76.43% better than inscribed ball. Meanwhile, the excluded proportions by IBCP in Step 4 is also shown in Table 1.

Table 1. The average proportions of points excluded by MVIE, inscribed ball and IBCP for all six types of data sets

Points excluded by, %	Data type					
	<i>mvnrnd</i>	<i>unifrnd</i>	<i>exprnd</i>	<i>evrnd</i>	<i>lognrnd</i>	<i>johnsrnd</i>
Inscribed ball	83.27	50.9	25.16	70.63	2.86	99.8
MVIE	97.88	51.42	26.43	77.85	79.29	99.86
IBCP	99.44	96.37	87.31	98.87	98.48	99.99

Table 2. The average computation time of 3D convex hull algorithms. The unit is second

Point amount	Algorithm	Data type					
		<i>mvnrnd</i>	<i>unifrnd</i>	<i>exprnd</i>	<i>evrnd</i>	<i>lognrnd</i>	<i>johnsrnd</i>
50,000	Quickhull	0.0324	0.0491	0.0433	0.0327	0.0308	0.0252
	Chan's	0.058	0.0541	0.0576	0.0321	0.0434	0.0368
	Proposed	0.0265	0.0409	0.042	0.0339	0.029	0.0229
100,000	Quickhull	0.0586	0.0863	0.0688	0.0559	0.0574	0.0492
	Chan's	0.0586	0.0863	0.0688	0.0559	0.0574	0.0492
	Proposed	0.0418	0.0854	0.0883	0.0438	0.0368	0.045
200,000	Quickhull	0.1057	0.1254	0.116	0.1032	0.1003	0.0913
	Chan's	0.1244	0.1453	0.186	0.1155	0.123	0.101
	Proposed	0.078	0.1511	0.1605	0.0768	0.0551	0.0834
1000,000	Quickhull	0.4354	0.456	0.4282	0.3948	0.4031	0.475
	Chan's	0.6556	0.5734	0.7356	0.456	0.591	0.6435
	Proposed	0.213	0.5912	0.563	0.3275	0.2693	0.3842
5000,000	Quickhull	2.0848	2.671	2.5931	2.2602	1.9391	2.4261
	Chan's	2.5623	3.1112	3.12	2.5308	2.245	2.63
	Proposed	0.9946	2.8878	3.0826	1.507	1.2115	1.065
10,000,000	Quickhull	4.1411	5.96	7.5421	4.2776	3.2922	4.073
	Chan's	5.343	6.432	8.156	5.661	4.367	4.8567
	Proposed	1.775	4.8958	5.929	2.3748	1.3712	1.834

Table 3. The peak memory usage of 3D convex hull algorithms. The data size is 1000,000. The unit is KB

Algorithm	Data type					
	<i>mvnrnd</i>	<i>unifrnd</i>	<i>exprnd</i>	<i>evrnd</i>	<i>lognrnd</i>	<i>johnsrnd</i>
Quickhull	874,492	874,492	874,492	874,492	874,492	874,492
Chan's	860,328	860,328	860,328	860,328	860,328	860,328
Proposed	484,352	484,352	484,352	484,352	484,352	484,352

3.3 The comparison of time consumption

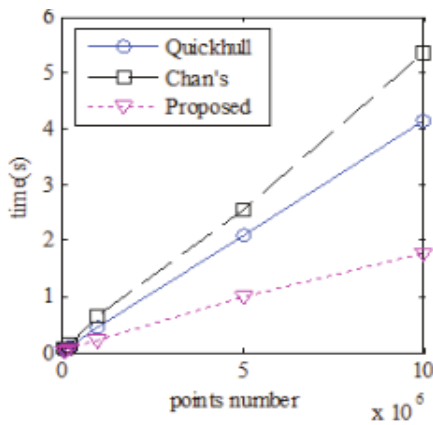
Different convex hull algorithms are used one by one to construct 3D convex hull, and compared the time consumption with our algorithm.

The proposed algorithm is faster than Chan's and Quickhull algorithms for all the six distributions of data sets. With the increasing of the point amount, it will save much computational time and perform more

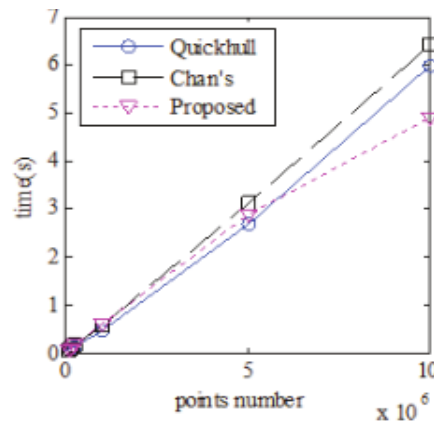
effectively. It only takes about 1.78 seconds to deal with 10,000,000 points for the *mvnrnd* distribution (the details are shown in Table 2). The computational time is less than Quickhull algorithm when the number of three-dimensional point set increases.

We also describe the time consumption in Table 2 as the curves according to different distribution types (see Fig.8). The curves tell us when the proportion of

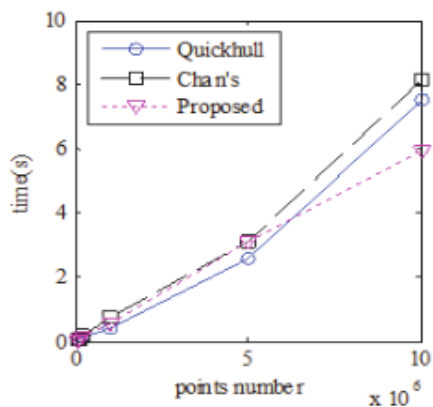
points excluded is over 50%, the proposed algorithm will faster than Quickhull and Chan's algorithm.



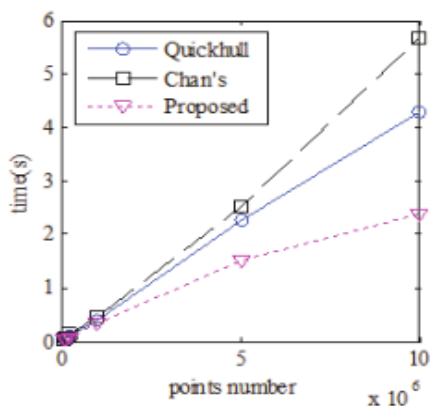
A mvnrd



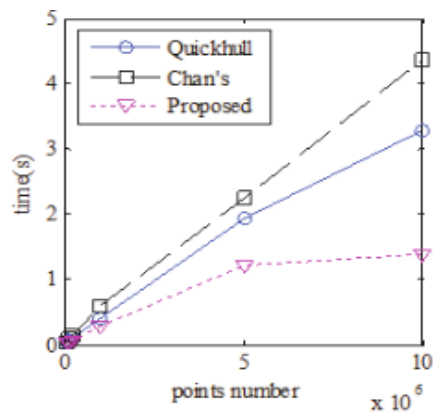
B unifrnd



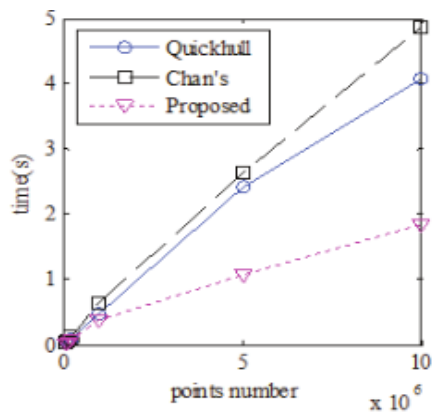
C exprnd



D evrnd



E logrnd



F Johnson

After analyzing the run time of all five steps in our algorithm, we know that the Steps 1, 2, 3, and 4 can be completed in $O(n)$ time. Since we use Quickhull in Step 5 to process the remaining points, the proposed algorithm runs in $O(n+(n-I)\log(n-I))$ time, where I is the number of non-vertex points excluded in IBCP. When the excluded proportion of points increases in IBCP, for example, the number of remaining points $(n-I)$ will be close to zero, the time complexity is nearly $O(n)$. Thus, our algorithm outperforms Quick-

hull in most cases. In Table 3, it shows that our algorithm can save over 50% of the peak memory space compared with the other algorithms. The major difference between our algorithm and Quickhull at the initial step is that the proposed algorithm eliminates most of interior points using an inscribed ellipsoid instead of a tetrahedron.

4. Conclusions

3D convex hull plays an important role in pattern recognition, clustering, data mining, image proces-

sing, virtual reality, and outlier detection, etc. In this paper, inspired by visual attention mechanism, an efficient algorithm to compute convex hull of 3D point set based on elastic ellipsoid is proposed. Firstly, some extreme points of a 3D point set will be obtained to construct IBCP, which is an initial estimation of the boundary of the point set. Secondly, the interior points in MVIE are removed. Finally, the remaining points outside IBCP are processed recursively by Quickhull algorithm and all vertexes of convex hull will be found. Experimental results show that the computational time of our algorithm is better than Chan's and Quickhull algorithms, and it can be applied to compute a convex hull in 3D massive data set.

Acknowledgements

This work is sponsored by Major Program of National Natural Science Foundation of China (Grant No.61190122), it is also supported by Scientific and Technological Research Program of Chongqing Municipal Education Commission (Grant No.KJ15012028 and No.KJ15012001).

References

- Xia, S., Xiong, Z., Luo, Y., Dong, L., Xing, C. (2015) Relative Density based Support Vector Machine. *Neurocomputing*, vol.149, p.p.1424-1432.
- Shahnewaz, S., Rahman, M.A., Mahmud, H. (2011) A Self Acting Initial Seed Selection Algorithm for K-means Clustering based on Convex-hull. *Proc. Conf. on Informatics Engineering and Information Science*, Kuala Lumpur, Malaysia, p.p.641-650.
- Pei, Y., Page, J., Pearce, G. (2015) A Methodology of Integrating Convex Hull into Data Mining for Simulation of Helicopter Autorotation. *Proc. Conf. on the 6th Australian International Aerospace Congress*, Singapore, p.p.409-414.
- Hoernig, M., Herrmann, M., Radig, B. (2015) Real-time Segmentation Methods for Monocular Soccer Videos. *Pattern Recognition and Image Analysis*, 25(2), p.p.327-337.
- Kim, B., Kim, K.J. (2012) Computing the Convex Hull for A Set of Spheres on A Gpu. *Proc. Conf. on the 11th ACM SIGGRAPH International Conference on Virtual-Reality Continuum and its Applications in Industry*, Singapore, p.p.345-345.
- Haynes, E.F., Taylor, M. (2014) An Assessment of Acoustic Contrast between Long and Short Vowels Using Convex Hulls. *The Journal of the Acoustical Society of America*, 136(2), p.p.883-891.
- Xia, S.y., Xiong, Z.y., He, Y., Li, K., Dong, L.m., Zhang, M. (2014) Relative Density-based Classification Noise Detection. *Optik*, 125(22), p.p.6829-6834.
- Graham, R.L. (1972) An Efficient Algorithm for Determining The Convex Hull of A Finite Planar Set. *Information processing letters*, 1(4), p.p.132-133.
- Jarvis, R.A. (1973) On the Identification of The Convex Hull of A Finite Set of Points in The Plane. *Information Processing Letters*, 2(1), p.p.18-21.
- Preparata, F.P., Hong, S.J. (1977) Convex Hulls of Finite Sets of Points in Two and Three Dimensions. *Communications of the ACM*, 20(2), p.p.87-93.
- Clarkson, K.L., Shor, P.W. (1989) Applications of Random Sampling in Computational Geometry, ii. *Discrete & Computational Geometry*, 4(1), p.p.387-421.
- Barber, C.B., Dobkin, D.P., Huhdanpaa, H. (1996) The Quickhull Algorithm for Convex Hulls. *ACM Transactions on Mathematical Software*, 22(4), p.p.469-483.
- Chazelle, B., Matousek, J. (1995) Derandomizing An Outputsensitive Convex Hull Algorithm in Three Dimensions. *Computational Geometry*, 5(1), p.p.27-32.
- Chan, T.M. (1996) Optimal Output-sensitive Convex Hull Algorithms in Two and Three Dimensions. *Discrete & Computational Geometry*, 16(4), p.p.361-368.
- Miller, R., Stout, Q.F. (1988) Efficient Parallel Convex Hull Algorithms. *IEEE Transactions on Computers*, 37(12), p.p.1605-1618.
- Amato, N.M., Preparata, F.P. (1993) An Nc Parallel 3d Convex Hull Algorithm. *Proc. Conf. on the 9th annual symposium on Computational geometry*, San Diego, CA, USA, p.p.289-297.
- Gupta, N., Sen, S. (2003) Faster Output-sensitive Parallel Algorithms for 3d Convex Hulls and Vector Maxima. *Journal of Parallel and Distributed Computing*, 63(4), p.p.488-500.
- Cao, T.T., Tang, K., Mohamed, A., Tan, T.S. (2010) Parallel Banding Algorithm to Compute Exact Distance Transform with The Gpu. *Proc. Conf. on the 2010 ACM SIGGRAPH symposium on Interactive 3D Graphics and Games*, Washington, USA, p.p.83-90.
- Stein, A., Geva, E., El-Sana, J. (2012) Cuda-hull: Fast Parallel 3d Convex Hull on The Gpu. *Computers & Graphics*, 36(4), p.p.265-271.

20. Chadnov R, V., Skvortsov A, V. (2004) Convex Hull Algorithms Review. *Proc. Conf. on the 8th Russian-Korean International Symposium on Science and Technology*, Tomsk, Russia, p.p.112-115.
21. Li, J., Duan, H. (2014) Novel Biological Visual Attention Mechanism via Gaussian Harmony Search. *Optik*, 125(10), p.p.2313-2319.
22. Xing, C., Xiong, Z., Zhang, Y., Wu, X., Dan, J., Zhang, T. (2014) An Efficient Convex Hull Algorithm Using Affine Transformation in Planar Point Set. *Arabian Journal for Science and Engineering*, 39(11), p.p.7785-7793.
23. Liu, R., Fang, B., Tang, Y.Y., Wen, J., Qian, J. (2012) A Fast Convex Hull Algorithm with Maximum Inscribed Circle Affine Transformation. *Neurocomputing*, 77(1), p.p.212-221.
24. Boyd, S., Vandenberghe, L. (2004) *Convex optimization*. Cambridge University Press: Cambridge.



Automatic Calibration of Computer Vision Based on RAC Calibration Algorithm

Lili Zhang

School of Physical Education, Langfang Teachers' College, Langfang 065000, Hebei, China

Dazhi Wang

Langfang Vocational and Technical College, Langfang 065000, Hebei, China

Abstract

In order to improve the precision of automatic calibration of computer vision, RAC calibration algorithm is used to realize automatic calibration. First, some briefly introduction of calibration method should be known, then the details of Mathematical model of the RAC calibration algorithm and the calibration process will be introduced. Finally, we use the simulation example to validate the performance of this method. The results of the experiments show that this method has realized the automatic calibration of computer vision. The fit of the measured value and the actual value is high, so this method has high application value.

Key words: COMPUTER VISION AUTOMATIC CALIBRATION RAC CALIBRATION