Multimed Tools Appl (2018) 77:31221–31237
https://doi.org/10.1007/s11042-018-6185-0

A fast convex hull algorithm inspired by human visual perception

Runzong Liu[1] · **Yuan Yan Tang**[2] · **Patrick P. K. Chan**[3]

Received: 13 December 2017 / Revised: 17 May 2018 / Accepted: 22 May 2018 /
Published online: 5 June 2018

Abstract This paper proposes a convex hull algorithm for high dimensional point set, which is faster than the well-known Quickhull algorithm in many cases. The main idea of the proposed algorithm is to exclude inner points by early detection of global topological properties. The algorithm firstly computes an initial convex hull of $2*d+2^d$ extreme points. Then, it discards all the inner points which are inside the inscribed ball of the initial convex hull. The other inner points are processed recursively according to the relationships of points and facets. Maximum inscribed circle affine transformations are also designed to accelerate the computation of the convex hull. Experimental results show that the proposed algorithm achieves a significant saving of computation time in comparison with the Quickhull algorithm in 3, 4 and 5 dimensional space. The space efficiency of the proposed algorithm is also demonstrated by experimental results.

Keywords Convex hull · Computational geometry · Affine transformation · Point pattern · High dimension

1 Introduction

The definition of convex hull of a d-dimensional point set is as follows: A set of points is defined to be convex if it contains the line segments connecting each pair of its points. The

✉ Runzong Liu
abslrz@cqu.edu.cn

[1] College of Computer Science, Chongqing University, Chongqing, 400030, People's Republic of China

[2] Faculty of Science and Technology, University of Macau, Macau, China

[3] South China University of Technology, Guangdong Sheng, China

convex hull of a set of points is the smallest convex set that contains the points [3] (see Fig.1).

Convex hull has applications to other geometric problems such as Delaunay triangulation and computation of the width and diameter of a point set. Besides, the convex hull algorithms also find various practical applications [5, 17, 24, 28]. For multimedia applications, wavelet [36] is a useful tool in the frequency Domain, while convex hull is a useful structure in the spatial domain, such as image processing[30], image registration[21], classification[10, 31], cluster analysis [18] and virtual reality [35]. Object detection [37], visual tracking [14–16] and pattern recognition [26, 38] are also important fields of multimedia analysis [25]. Convex hull algorithms are useful in these fields as well [4, 20, 23, 32–34].

The common architecture of convex hull algorithms is shown in Fig. 2, with each component defined as follows:

- Initial convex hull: The initial convex hull is a simplified d-polytope. It can be a starting point (Granham scan algorithm [12]), or points for partitioning (Andrew algorithm [1], Quickhull algorithm [3]). The initialization is important for the algorithm efficiency.
- Point location: Point location can be sorting the points (Graham scan algorithm), or locating the points to specific partitions (Andrew algorithm, Quickhull algorithm). For general dimensions, when a cone of new facets is created, it builds new outside sets from the outside sets of the visible facets. This point location process locates a visible new facet for each remaining point.
- Point comparison: Point comparison is to compare the query point with some threshold to decide whether the point is a vertex of the convex hull. The thresholds might be different for different algorithms. For general dimensions, the hyperplane defines a halfspace of points that have negative distances from the hyperplane. If the distance is positive, the point is above the hyperplane. This process is called as point comparison.
- Renew the convex hull: The renewal of a convex hull can be made by adding a new vertex to the original convex hull vertex set (Graham scan algorithm), or by merging several convex hull vertex subsets into a larger convex hull vertex subset or the whole convex hull vertex set (Andrew algorithm, Quickhull algorithm).

The following terminologies are some basics for discussion of convex hull problem in general dimensions.

A d-dimensional convex polyhedron is the intersection of a finite number m of non-redundant half-spaces $H = H_1, H_2, ..., H_m of I R^d$ [2]. A bounded convex polyhedron is called a $polytope$. The smallest convex set containing point set S is a polytope $conv(S)$

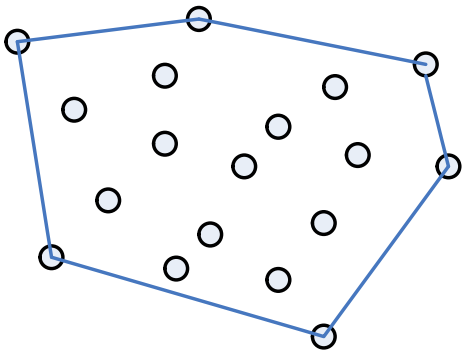

Fig. 1 The definition of convex hull. In the figure, the polygon is the convex hull of the point set

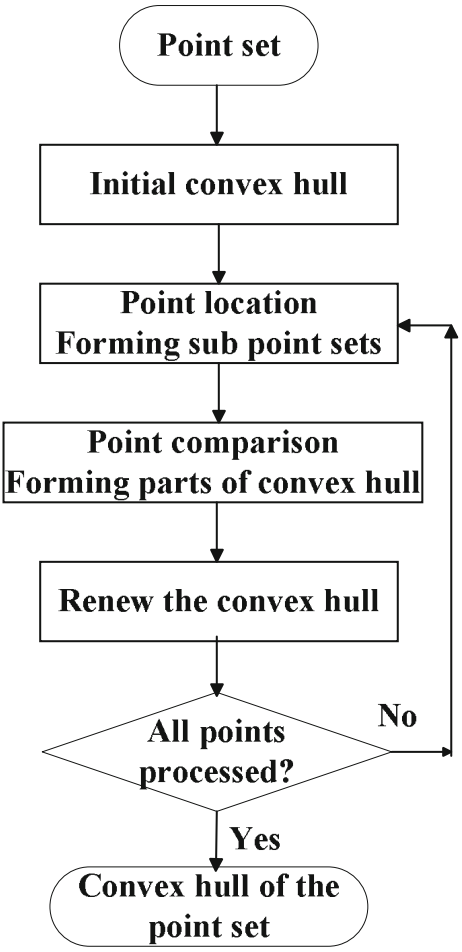

Fig. 2 The common architecture of a convex hull algorithm

called the convex hull of S. The extreme points of S are the vertices of the convex hull $conv(S)$(or equivalently, the set of points $p \in S$ with conv(S - { p}) $\neq$ conv(S)).

Here, we assume that S is in non-degenerate position, i.e., no $d + 1$ points of S lie in a common hyperplane. Such non-degeneracy can easily be simulated with impunity using standard perturbations techniques [11]. Non-degeneracy ensures that the convex hull of any subset of S is a simplified polytope[29].

Let P be a simplified d-polytope, let V be the vertex set of P, and let $n = |V|$. The (d-1) dimensional faces of P are called as facets, and the (d-2)-dimensional faces are called as ridges. Every facet is uniquely identified by d vertices, while every ridge can be identified by $d - 1$ vertices in V. Each ridge is the intersection of the vertices of two neighboring facets. We represent a d-dimensional convex hull by its vertices and (d-1)-dimensional facets. Therefore, convex hull can be represented by a set of facets and a set of adjacency lists giving the neighbors and vertices. In R^3 and general position, facets are triangles and ridges are edges.

Let p be some point in R^d in non-degenerate position with respect to V. A facet F of P is called as visible from p if the hyperplane spanned by F separates P and p. Otherwise, the

facet F is called as obscured. A face G of P is called visible from p if it is only contained in facets of P that are visible from p. Obscured faces are defined analogously. The face G is called as a $horizon$ face with respect to some point x if it is contained in some visible and some obscured facet. Ridges contained in $horizon$ face are called as $horizon$ ridges.

The terminologies allow a convenient characterization of the facial structure of the polytope $P' = conv(PUx)$ in terms of the faces of P [29].

2 Related work

The first algorithm to compute convex hull appeared in the pioneering 1953 paper of Motzkin et al [22]. Numerous convex hull algorithms have been developed over the past decades. Most of them are designed particularly for the cases of two dimensions [1, 6, 12] in the consideration of efficiency. Some of them are capability to compute convex hull in space of 3 dimensions [6, 27]. Only a few algorithms can be extended to the cases of higher dimensions [3, 7].

Barber et al. [3] developed an efficient convex hull algorithm, Quickhull, which was built upon the algorithm of Clarkson and Shor [9]. Quickhull works in the space of points and convex hulls instead of the dual space of half-spaces and polytopes. The biggest advantage of Quickhull algorithm is that it processes the furthest point of an outside set, instead of a random point. Due to its efficiency and capability in general dimensions, Quickhull is currently the most popular convex hull algorithm. However, there are still many redundant operations in Quickhull algorithm.

In this paper, we propose a convex hull algorithm inspired by the early detection of global topological properties in visual perception [8]. The proposed algorithm excludes massive inner points directly by an initial inscribed ball. It deals with high dimensional point sets by the divide and conquer approach with a single processor. We will compare our proposed algorithm with Quickhull in the experiments.

3 The proposed convex hull algorithm for high dimensions

Without loss of generality, we assume that the considering space is of dim dimensions.

3.1 Point location and point comparison of the proposed algorithm

For point location, the proposed algorithm applies the early detection of global topological properties in visual perception [8]. This important work [8] about visual attention proved that extraction of global topological properties is a basic factor in perceptual organization. Three experiments showed that when a perceptual stimulus is presented under impoverished visual conditions, figure and ground achieve some measure of differentiation, although the detailed structure of the stimulus remains vague and amorphous. This demonstrates that the global topological properties are extracted earlier than the detailed structures.

Applying this discovery of perceptual organization, our algorithm firstly computes an initial convex hull of $2 * dim + 2^{dim}$ extreme points. Then, it excludes all the inner points inside the inscribed ball of the initial convex hull. An example of the inscribed ball of the initial convex hull of a point cloud is shown in Fig. 3. The other inner points are processed based on the Beneath-Beyond Theorem [13] similar to the two dimensional convex hull algorithm [19] in our earlier publication.

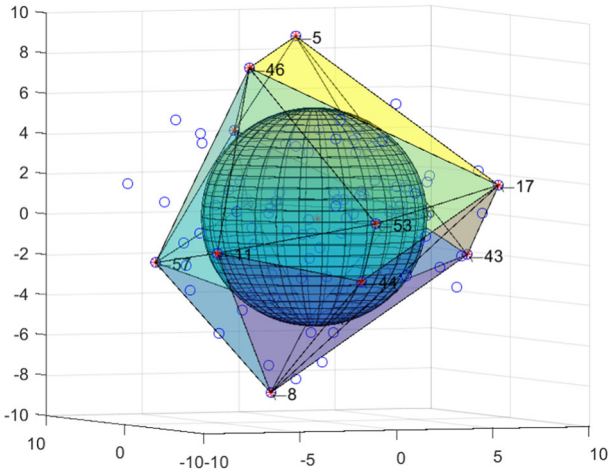

Fig. 3 The inscribed ball of the initial convex hull of a point cloud

3.2 Maximum inscribed circle affine transformation

The maximum inscribed circle affine transformation (MICAT) is also applied here to solve the narrow distribution problem. In some cases, the shape of the convex hull of a point set is narrow or thin. In these cases, the radius length of the inscribed ball of the initial convex hull can be enlarged by particular affine transformations. Thus, more inner points can be included in the inscribed ball and be excluded at the early stage.

The two-dimensional affine transformation discussed here can be defined as:

$$\begin{bmatrix} y_1 \\ y_2 \end{bmatrix} = \begin{bmatrix} k_{11} & k_{12} \\ k_{21} & k_{22} \end{bmatrix} * \begin{bmatrix} x_1 - ox_1 \\ x_2 - ox_2 \end{bmatrix} \tag{1}$$

$$\begin{aligned} k_{11} &= t * \cos^2\theta + \sin^2\theta \\ k_{12} &= \sin\theta * \cos\theta * (t-1) \\ k_{21} &= \sin\theta * \cos\theta * (t-1) \\ k_{22} &= t * \sin^2\theta + \cos^2\theta \end{aligned}$$

where ox is the coordinate vector of the centroid O. θ is the stretching direction which passes through the centroid O. t is the stretching intensity. Since $\varrho(\theta, t)$ is equal to the composition of $trans(O)rot(\theta)scale_y(t)$ with O is the centroid, the coefficient k_{ij} could be derived by the corresponding coefficient of $rot(\theta)scale_y(t)$. The $trans(O)$ is a translation which makes the centroid of the point set locate at the origin. $rot(\theta)$ is a rotation which makes the stretching direction along the y axis. $scale_y(t)$ is a stretching along the y axis with intensity t.

For computing MICAT, we have to specify θ and t to make the polygon have two intersections with its inscribed circle. In order to find a new intersection while keeping the original intersection unchanged, the stretching direction φ is the direction from the centroid O to the intersection D of the inscribed circle and the polygon (see Fig. 4). Thus, we only need to calculate all the stretching intensities t_i which satisfy that the inscribed circle has a new intersection with the i_{th} edge of the polygon while keeping the original intersection unchanged. The minimum of t_i is the answer for t, because it can keep the circle inscribed to the polygon under the affine transformation while others can not.

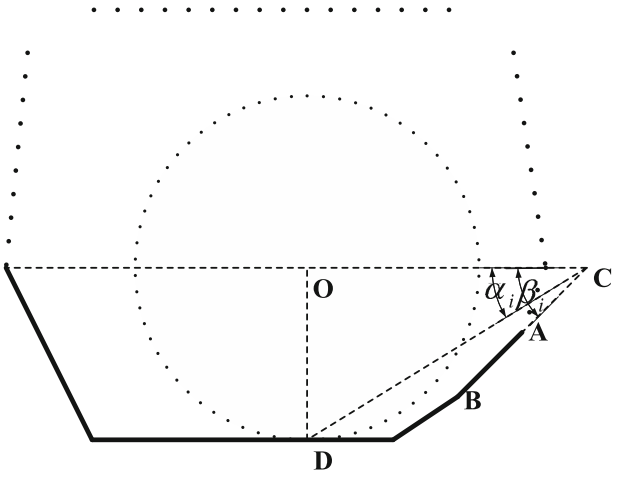

Fig. 4 A part of a polygon and its inscribed circle. OC is orthogonal to OD

The t_i can be derived by the equation:

$$t_i = \sqrt{\cot^2 \alpha_i - \cot^2 \beta_i} \tag{2}$$

where α_i and β_i are explained by Fig. 4. In Fig. 4, O is the centroid of the polygon, and D is the intersection of the initial inscribed circle and the polygon; AB is the i_{th} edge of the polygon; The point C is the intersection of line AB and the direction through O orthogonal to OD; α_i is the angle of the lines OC and CD; β_i is the angle of the edge AB and the line OC.

The procedure of calculating t_i can be referred to the literature [19].

Then, MICAT is the affine transformation

$$\varrho(\varphi, t), t = \arg\min(t_i = \sqrt{\cot^2 \alpha_i - \cot^2 \beta_i}) \tag{3}$$

The left part of Fig. 5 shows a 3D point set and the inscribed ball of its initial convex hull before MICAT, while the right part of Fig. 5 describes the same point set after the application of MICAT. We can see that more points can be excluded by the inscribed ball of the initial convex hull under MICAT.

3.3 Visual-attention-imitation convex hull algorithm with maximum inscribed circle affine transformation

The input of the proposed algorithm is the point set S of size m in dim dimensional space, while the output is the convex hull of S. The eight steps of the algorithm are as follows:

Step 1. Searching extreme points: The algorithm finds $2 * dim + 2^{dim}$ extreme points of the point set;

Step 2. The initial convex hull: We apply the Quickhull algorithm to calculate the convex hull of the extreme points in step 1, and it is the initial convex hull of the whole point set (see Fig. 6);

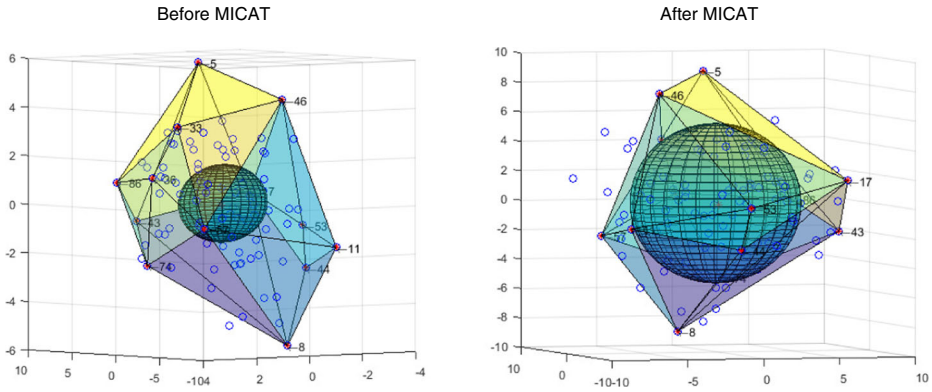

Fig. 5 A 3D point set and the inscribed ball of its initial convex hull before and after MICAT

Step 3. Applying affine transformation: Both the shape of the initial convex hull and the size of the point set are checked. If the point set is large enough and its initial convex hull is too thin , the maximum inscribed circle affine transformation (MICAT) is applied;

Step 4. Threshold A: An inscribed ball of the initial convex hull is calculated, whose center locates at the centroid of the whole point set; The points inside the inscribed ball are apparently not the vertices of the output convex hull, and they should be removed;

Step 5. Point location: The rest points are processed recursively according to the location relationships of the points and the facets of the initial convex hull. If a point is inside the initial convex hull, it should be deleted, otherwise, the algorithm puts it into the outside point set of a facet who is visible from it;

Step 6. Threshold B and C: For the outside point set of each facet of the current convex hull, the algorithm searches the farthest point from the facet. This farthest point is also a new vertex of the output convex hull. This new vertex and each

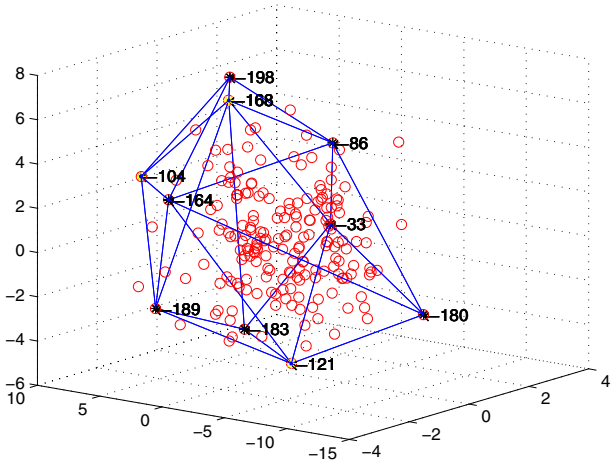

Fig. 6 The initial convex hull of a point cloud

of its horizon ridges determine a new facet. And all these new facets construct a cone. The shortest distance from a facet of the new cone to the centroid is used for Threshold B. Any point whose distance from the point to the centroid is less than Threshold B is inside the cone, and it should be removed. Then, the algorithm deletes the visible facets from the new vertex. Any point who has passed Threshold B will be checked whether it is visible by the facets of the new cone. This checkup is called as Threshold C. Each of the rest points who has passed Threshold C will be put into the outside point set of a current facet which is visible from it;

Step 7. Processing points recursively: If all the points are processed, the current convex hull is an equivalent of the output convex hull. Otherwise, the rest points are recursively processed by step 6;

Step 8. The reverse MICAT: The algorithm checks whether the MICAT is applied. If it is applied, the reverse MICAT should be applied on the current convex hull. The current convex hull is the output convex hull of the algorithm.

Below, we will discuss some steps of the proposed algorithm in detail.

The detail procedure of step 1 is as follows: The following $2*dim+2^{dim}$ extreme points of the point set can be found by simple calculations: point with maximum coordinate of some dimension; point with minimum coordinate of some dimension; points that maximize the functions f_j shown in 4.

$$\begin{aligned} f_j &= \sum_{i=1}^{\dim} a_{ij}x_i,\ j = 1, 2, 3, \cdots, 2^{\dim} \\ j-1 &= \sum_{i=0}^{\dim-1} b_{ij}2^i, \\ a_{ij} &= 2b_{ij} - 1. \end{aligned} \tag{4}$$

In step 3, for a large point set of a thin shape, MICAT is applied to accelerate the computation. For simplification, we do not design n dimensional MICAT for space of n dimensions. Instead, we apply the two dimensional MICAT [19] on n dimensional point set for every space of each pair of dimensions. Besides, the saving of computation by MICAT depends on how many additional points can be excluded by Threshold A, and this depends on the stretching intensity t and the size of the point set. The stretching intensity t under MICAT is a property of the shape of a point set. Although the application of MICAT can exclude as many inner points as possible by Threshold A and thus save lots of computation, the computation of MICAT itself should be compensated. Therefore, before applying MICAT on the whole point set, for an unknown point pattern, we can calculate and check the stretching intensity t for MICAT on the initial convex hull mentioned above for every space of each pair of dimensions.

In step 4, the radius of the inscribed ball of the initial convex hull is used to exclude inner points. If the distance from a point to the centroid is shorter than the length of the radius, the point is inside the convex hull and it will be neglected. This threshold is named as Threshold A in our earlier publication [19] for convex hull in a plane. It is quite good that point location is not needed for Threshold A, while it is a basic and time-consuming query process for every point for other convex hull algorithms.

After excluding the inner points inside the inscribed ball, the step 6 of the algorithm calls a sub algorithm to recursively process the rest points using the spherical threshold B and the threshold C according to the Beneath-Beyond Theorem [13]. A summary

of the sub algorithm according to the Beneath-Beyond Theorem is given by a table of Algorithm 1.

The detail of step 1 in Algorithm 1 is as follows: For a checking point p, if there is some facet visible from it, the algorithm records all the visible facets of p and puts p in the outside set of one of its visible facets. If no facet of the initial hull is visible from point p, point p is inside the initial convex hull, and it will be neglected.

The spherical threshold B only needs one comparison of two distances to check whether a point is inside a cone in space of 3 or higher dimensions. If the threshold is not applied, every facet of the cone should be checked to find out whether a point is inside the cone. If there is no visible facet for a checking point, this point is inside the convex hull and should be neglected.

Algorithm 1 The algorithm calculates the convex hull of the point set P according to the beneath-beyond theorem

```
1: Calculate the outside sets of facets F of the initial convex hull.
2: while there is a facet F with a non-empty outside set do
3:     Select the furthest point p of F's outside set.
4:     Calculate the set of horizon ridges H of p.
5:     for each ridge R in H do
6:         Create a new facet F' from R and p.
7:         Calculate the distance fd from the centroid O to the new facet F'.
8:     end for
9:     Calculate the shortest one MINfd from the distance set of fd.
10:    for each point q in the outside sets of the visible set V do
11:        Calculate the distance qd from point q to the centroid O.
12:        if qd is less than MINfd then
13:            Threshold B: Delete point q and turn to check next point.
14:        end if
15:        Threshold C:
16:        for each new facet F' do
17:            if q is above a new facet F' then
18:                Assign q to the outside set of F'; Break.
19:            end if
20:        end for
21:    end for
22:    Renew the facets F and their outside sets.
23: end while
```

4 Experimental results

Experiments were carried out on a PC with Intel Pentium 2.40 G Hz CPU and Windows 10 operating system using Matlab9.0.0.341360

Six types of data sets were built in experiments to compare the proposed algorithm (VAICH-M) with Quickhull algorithm, which were generated by the following functions in Matlab9.0.0.341360 shown in Fig. 7. Function mvnrnd returns a matrix S of random vectors chosen from the multivariate normal distribution with mean MU and covariance SIGMA. The functions unifrnd, exprnd, evrnd, lognrnd, and johnsrnd return a matrix S of random

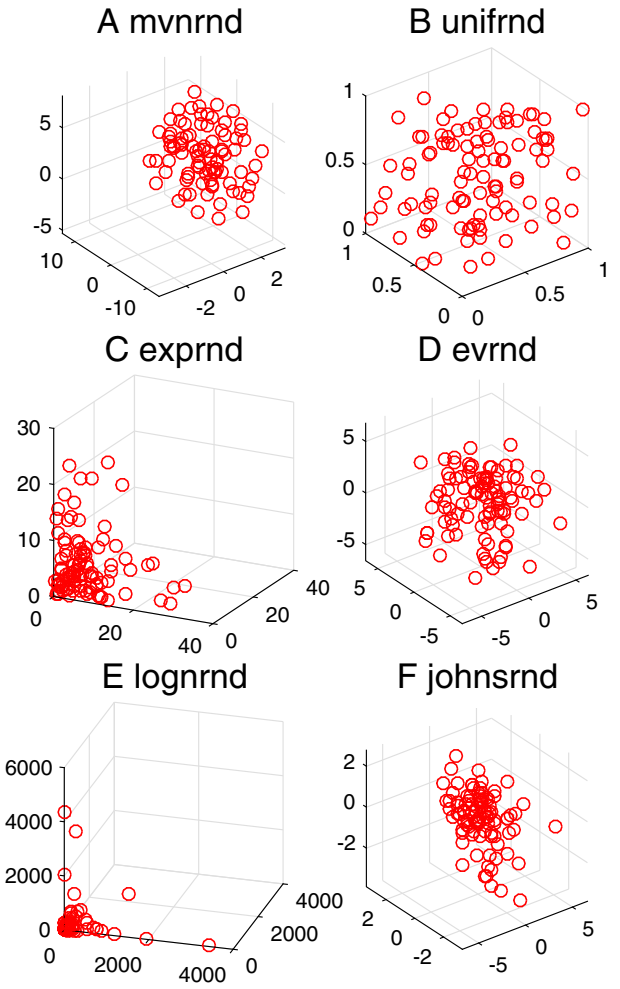

Fig. 7 Six types of testing data generated by Matlab7.4.0

vectors generated from the uniform distribution, the exponential distribution, the extreme value distribution, the lognormal distribution, and the distribution in the Johnson system respectively.

4.1 Comparison with Quickhull in six random cases in space of 3, 4, and 5 dimensions

In Table 1, VAICH-O represents the proposed visual-attention-imitation convex hull algorithm without the application of MICAT. From Table 1, we can see that VAICH-O is much faster than Quickhull algorithm, for all the six types of experiment data. We can also see that the larger the point amount is, the more saving of computation time VAICH-O achieves. For example, for the data of unifrnd type in 3 dimensional space, when the point amount is 10000, VAICH-O is about two times faster than Quickhull; when the point amount is

Table 1 Average computation time of convex hull algorithms for point sets in 3 dimensional space

Dim	Point amount	Algorithm	Data type					
			mv	un	ex	ev	lo	jo
3	10000	Quickhull	1.00	2.13	1.22	0.71	0.78	0.49
		VAICH-O	0.87	0.78	0.92	0.39	0.57	0.03
3	20000	Quickhull	2.16	6.08	2.86	1.39	1.84	0.77
		VAICH-O	1.60	1.49	1.51	0.56	1.13	0.04
3	40000	Quickhull	3.42	13.69	6.94	3.00	3.86	1.57
		VAICH-O	2.44	2.74	3.31	1.13	2.18	0.03
4	10000	Quickhull	7.92	53.62	19.92	7.05	10.40	1.37
		VAICH-O	4.04	7.72	5.29	2.59	2.04	0.26
4	20000	Quickhull	18.87	167.53	54.81	15.84	27.34	2.37
		VAICH-O	7.06	13.87	9.43	4.73	4.40	0.21
4	40000	Quickhull	49.36	542.46	140.56	36.43	59.73	5.50
		VAICH-O	13.32	25.64	18.77	9.23	7.58	0.32
5	1000	Quickhull	4.05	16.54	8.09	3.56	4.68	0.96
		VAICH-O	2.49	7.25	4.07	2.59	2.14	0.82
5	2000	Quickhull	11.37	63.64	27.26	10.21	162.01	2.53
		VAICH-O	4.82	16.61	8.26	4.79	3.72	1.41
5	4000	Quickhull	32.27	233.81	76.96	28.03	753.19	4.25
		VAICH-O	10.19	35.41	15.93	8.88	6.87	1.54

The unit is second. VAICH-O represents the proposed visual-attention-imitation convex hull algorithm without the application of MICAT. The abbreviations of mv, un, ex, ev ,lo and jo represent mvnrnd, unifrnd, exprnd, evrnd, lognrnd and johnsrnd respectively

20000, VAICH-O is about three times faster; and when the point amount is 40000, VAICH-O is about four times faster. This is because that when the size of the data of the same type becomes larger, the density of points increased, thus, more inner points can be excluded quickly by Threshold A and Threshold B. Besides, for all the six types of testing data except the lognrnd type, VAICH-O is about two or three times faster than Quickhull algorithm, when the point amount is 40000 in the experiment. For the type of johnsrnd, VAICH-O is about 50 times faster than Quickhull algorithm.

From Table 1, we can also see that VAICH-O achieves more saving of computation time in 4 dimensional space than that in 3 dimensional space. For example, for the unifrnd type, when the point amount is 10000, VAICH-O is about 10 times faster than Quickhull, while it is only about two times faster in space of 3 dimension. This is because that the amount of operations for processing a 4 dimensional point is much larger than that in space of 3 dimensions. Thus, when the same amount of inner points are excluded by Threshold A or B, much more computation time are saved. This is also proved by the experimental results in space of 5 dimensions. Both Quickhull and VAICH-O spend more time to deal with one thousand points in 5 dimensional space than to that deal with ten thousand points of the same type in 3 dimensional space. Nevertheless, for all the six types of data in space of 3, 4 and 5 dimensions, VAICH-O is several times faster than Quickhull.

However, since the method of point comparison of Threshold C of VIACH-O is the same as that of Quickhull algorithm, the complexity of VAICH-O is the same as the complexity

of Quickhull algorithm. If the balance conditions hold, Quickhull runs on an input of size n with r processed points in time $O(n \log r)$ for $d \leq 3$ and $O(nf_r/r)$ for $d \geq 4$ (f_r is the maximum number of facets for r vertices) [3].

4.2 Threshold efficiency analysis in six random cases

In this experiment, we investigate threshold efficiency in six random cases for point set of size 10000 in space of 3 dimensions. The data was stretched along y (the second) coordinate axis with intensity t before being processed by the proposed algorithm. It is shown that about 50% of points were excluded by Threshold A for all the six types of testing data except the exprnd type and the lognrnd type in Table 2. For the type of johnsrnd, nearly all the points were excluded by Threshold A. For the type of mvnrnd, if MICAT is applied, more than 65% of points can be excluded by Threshold A, while it is only about 36% without the application of MICAT. Thus, the time cost of VAICH with MICAT is only about half of that without MICAT in this case.

We also investigate the situations when stretching is applied in prior on the data. From Tables 3 and 4, we can see that if the stretching intensity enhanced (the point set became narrower), the proportion of points excluded by Threshold A dropped dramatically without the application of MICAT. However, when the stretching intensity was enlarged to 4, Threshold A with MICAT could still exclude about 40% of points, for all the six types of testing data except the exprnd type and the lognrnd type. When the stretching intensity was enlarged to 1.5, the application of MICAT achieves considerable saving of computation time, for all the six types of testing data except the johnsrnd type.

The main advantage of the proposed algorithm is that Threshold A can be used to exclude inner points quickly. However, the worst case of the algorithm may be that a set of points

Table 2 Proportions of points excluded by Threshold A of VIACH with the application of MICAT (V-M) and by Threshold A by VAICH without the application of MICAT (V-O), and the corresponding convex hull computation time of V-M and V-O in 3 dimensional space when the point amount is 10000

Stretching	Threshold A	Data type					
intensity		mvnrnd	unifrnd	exprnd	evrnd	lognrnd	johnsrnd
1	V-O Prop.	36.11%	46.37%	20.58%	65.68%	2.21%	97.87%
1	V-M Prop.	65.46%	46.37%	20.58%	67.10%	2.21%	97.87%
1	V-O time	0.8585	0.9895	1.0199	0.4138	0.7241	0.0249
1	V-M time	0.4622	0.9888	1.0203	0.4168	0.7403	0.0258
1.5	V-O Prop.	25.30%	30.85%	13.49%	50.20%	1.40%	96.08%
1.5	V-M Prop.	57.81%	37.62%	18.46%	61.73%	2.13%	96.42%
1.5	V-O time	1.2739	1.8873	1.5042	0.8742	1.0877	0.0589
1.5	V-M time	0.6975	1.6097	1.3842	0.7873	1.0638	0.0597
4	V-O Prop.	8.01%	11.04%	4.59%	16.22%	0.42%	68.56%
4	V-M Prop.	43.12%	45.28%	20.71%	45.94%	2.95%	82.28%
4	V-O time	1.039	2.3533	1.816	1.3092	1.1387	0.3091
4	V-M time	0.6440	1.4891	1.4563	0.9054	0.9840	0.2177

The abbreviation 'Prop.' means the proportion.

Table 3 Time of computation of convex hull by VIACH with the application of MICAT (V-M) and by VAICH without the application of MICAT (V-O) in 3 dimensional space

Stretching intensity	1		2		4	
Point set size	V-O	V-M	V-O	V-M	V-O	V-M
100	0.0087	0.0095	0.0109	0.0114	0.0115	0.0113
200	0.0133	0.0133	0.0160	0.0160	0.0174	0.0161
400	0.0141	0.0149	0.0237	0.0211	0.0254	0.0207
800	0.0238	0.0231	0.0315	0.0243	0.0415	0.0278
1600	0.0244	0.0236	0.0408	0.0354	0.0635	0.0444

The data type is johnsrnd. The unit of the computation time is second

locate in a narrow annulus or a ball with a large hole. In this case, Threshold A makes no use.

4.3 Efficiency comparison between VAICH with MICAT and VAICH without MICAT

The relationship between the point set size and the efficiency of MICAT is investigated as well as the relationship between the stretching intensity and the time saving by the application of MICAT. In this experiment, the data was stretched along y (the second) coordinate axis with intensity t before being processed by the proposed algorithm. The experimental results in 3 dimensional space are shown in Table 3 while the results in 4 dimensional space are shown in Table 4. In the two tables, the point set size grows at a rate of two times from top to down, while the stretching intensity grows at a rate of two times from left to right. The time saving always increases with the size of the point set while it only increases with the stretching intensity t in some extent.

However, we should also notice that when the size of a point set is less than 200, the application of MICAT has no advantage for point set in 3 or 4 dimensional space. This demonstrates that the cost of the computation of MICAT and its application on the point set may be over the saving of computation by MICAT for VAICH. Based on the experimental results shown in Table 2, Tables 3 and 4, we could choose a threshold for t, e.g. $t > 1.5$,

Table 4 Time of computation of convex hull by VIACH with the application of MICAT (V-M) and by VAICH without the application of MICAT (V-O) in 4 dimensional space

Stretching intensity	1		2		4	
Point set size	V-O	V-M	V-O	V-M	V-O	V-M
100	0.0383	0.0400	0.0422	0.0436	0.0430	0.0431
200	0.0680	0.0679	0.0778	0.0753	0.0653	0.0645
400	0.0960	0.0948	0.1168	0.1115	0.1100	0.1077
800	0.1335	0.1326	0.1642	0.1545	0.1669	0.1699
1600	0.1656	0.1622	0.2656	0.2539	0.2892	0.2404

The data type is johnsrnd. The unit of the computation time is second

Table 5 Peak memory usage of convex hull algorithms when the data size is $2 * 10^5$ in 3 dimensional space

Algorithm	Data type					
	mvnrnd	unifrnd	exprnd	evrnd	lognrnd	johnsrnd
Quickhull	86808	118840	170510	96038	103390	81682
VAICH-M	69062	68571	71659	68275	77719	75207
saving	20.44%	42.30%	57.97%	28.91%	24.83%	7.93%

The unit is 10 KB. The last line reports the percentages of memory space saved by VAICH-M compared to Quickhull

and another threshold for the size of point set, e.g. 1000. If t is more than 1.5 and the size of the point set is more than 1000, we apply MICAT on that space; otherwise we do not apply MICAT.

4.4 Space efficiency comparison between Quickhull and VAICH-M

Table 5 also shows that VAICH-M uses less memory space than Quickhull algorithm. And it can save over 20% memory space compared to Quickhull algorithm for all the six data types except the johnsrnd type, when the point amount is $2 * 10^5$. Especially for the data of exprnd type, more than 50% memory space is saving. In fact, both Quickhull and VAICH-M are recursive algorithms which return corresponding parts of convex hull for given subsets, but the subsets of VAICH-M are smaller than those of Quickhull. It is because that we consider $2 * dim + 2^{dim}$ vertices for the initial convex hull of VAICH-M, while only dim vertices are considered for that of Quickhull. Therefore, many points are excluded by the inscribed ball of the initial convex hull of VAICH-M and they will not be included in the subsets, while no point can be excluded by the initial convex hull of Quickhull.

5 Conclusions

Inspired by early detection of global topological properties in human visual perception, we propose a convex hull algorithm for point set in space of high dimensions, which is an extension of the two dimensional visual-attention-imitation convex hull algorithm [19]. The main idea behind the proposed algorithm is using an important global topological structure, the inscribed ball of an initial convex hull, to exclude most inner points at the very beginning. Thus, the algorithm can save much computation of point location and point comparison. Experimental results show that visual-attention-imitation convex hull algorithm (VIACH) has better performance than Quickhull algorithm in 3, 4 and 5 dimensional space. However, the worst case of the algorithm may be that a set of points locate in a narrow annulus or a ball with a large hole. When recognizing the convex hull of a narrow and large point set, Maximum Inscribed Circle Affine Transformation (MICAT) is also applied to make VIACH faster. The virtue of MICAT in 3 and 4 dimensional space has been demonstrated by the experimental results. The proposed algorithm can enhance the efficiency of analysis of high dimensional data, collision detection of objects, and other applications of convex

hull. The paralleled computing version and the incremental learning version of the proposed algorithm are topics of further studies.

Acknowledgments This work was financially supported by Project No.106112016CDJXY180002, 0903005203327, 02160011044104 supported by the Fundamental Research Funds for the Central Universities. This work was also supported by the Research Grants of MYRG2015-00049-FST, MYRG2015-00050-FST, RDG009/FST-TYY/2012; 008-2014-AMJ from Macau.

References

1. Andrew A (1979) Another efficient algorithm for convex hulls in two dimensions. Inf Process Lett 9:216–219
2. Avis D, Bremner D, Seidel R (1997) How good are convex hull algorithms? Comput Geom 7(5):265–301
3. Barber CB, Dobkin D, Huhdanpaa H (1996) The quickhull algorithm for convex hulls. ACM Trans Math Softw 22:469–483
4. Bo C, Wang D (2016) Online object tracking based on convex hull representation. In: 2016 IEEE 22nd International Conference on Parallel and Distributed Systems (ICPADS), pp 1221–1224
5. Buliung RN, Kanaroglou PS (2006) A gis toolkit for exploring geographies of household activity/travel behavior. J Transp Geogr 14(1):35–51
6. Chan TM (1996) Optimal output-sensitive convex hull algorithms in two and three dimensions. Discret Comput Geom 16(4):361–368
7. Chand DR, Kapur SS (1970) An algorithm for convex polytopes. J ACM (JACM) 17(1):78–86
8. Chen L (1982) Topological structure in visual perception. Science 218(4573):699–700
9. Clarkson KL, Shor PW (1989) Applications of random sampling in computational geometry, ii. Discret Comput Geom 4(1):387–421
10. Ding S, Nie X, Qiao H, Zhang B (2017) A fast algorithm of convex hull vertices selection for online classification. IEEE Trans Neural Netw Learn Syst PP(99):1–15
11. Edelsbrunner H (1987) Algorithms in Combinatorial Geometry. Springer-Verlag, New York
12. Graham R (1972) An efficient algorithm for determining the convex hull of a finite planar set. Inf Process Lett 1:132–133
13. Grnbaum B (1963) Measures of symmetry for convex sets, Convexity Proceedings of Symposia in Pure Mathematics American Mathematical Society, pp 233–270
14. He Z, Cui Y, Wang H, You X, Chen CLP (2015) One global optimization method in network flow model for multiple object tracking. Knowl-Based Syst 86(C):21–32
15. He Z, Li X, You X, Tao D, Tang Y (2016) Connected component model for multi-object tracking. IEEE Trans Image Process Publ IEEE Signal Process Soc 25(8):3698
16. He Z, Yi S, Cheung YM, You X, Tang Y (2017) Robust object tracking via key patch sparse representation. IEEE Trans Cybern 47(2):354–364
17. Khosravani HR, Ruano AE, Ferreira PM (2016) A convex hull-based data selection method for data driven models. Appl Soft Comput 47:515–533
18. Liparulo L, Proietti A, Panella M (2015) Fuzzy clustering using the convex hull as geometrical model. Adv Fuzzy Syst 2015:39–51
19. Liu RZ, Fang B, Tang Y, Wen J, Qian J (2012) A fast convex hull algorithm with maximum inscribed circle affine transformation. Neurocomputing 77:212–221
20. Marin G, Dominio F, Zanuttigh P (2016) Hand gesture recognition with jointly calibrated leap motion and depth sensor. Multimed Tools Appl 75(22):1–25
21. Minhas R, Wu J (2007) Invariant feature set in convex hull for fast image registration. In: 2007. ISIC. IEEE International Conference on Systems, Man and Cybernetics. IEEE, pp 1557–1561
22. Motzkin TS, Raiffa H, Thompson G, Thrall RM (1953) The double description method
23. Mousse MA, Motamed C, Ezin EC (2017) People counting via multiple views using a fast information fusion approach. Multimed Tools Appl 76:1–19
24. Murtagh F (1992) A new approach to point-pattern matching. Publ Astron Soc Pac 104(674):301–307

25. Niu L, Zhou W, Wang D, He D, Zhang H, Song H (2016) Extracting the symmetry axes of partially occluded single apples in natural scene using convex hull theory and shape context algorithm. Multimed Tools Appl 76(12):1–15
26. Ou W, You X, Tao D, Zhang P, Tang Y, Zhu Z (2014) Robust face recognition via occlusion dictionary learning. Pattern Recogn 47(4):1559–1572
27. Preparata FP, Hong SJ (1977) Convex hulls of finite sets of points in two and three dimensions. Commun ACM 20(2):87–93
28. Renold AP, Chandrakala S (2017) Convex-hull-based boundary detection in unattended wireless sensor networks. IEEE Sens Lett 1(4):1–4
29. Seidel R (1991) Small-dimensional linear programming and convex hulls made easy. Discret Comput Geom 6(1):423–434
30. Szczypiński P, Klepaczko A (2010) Automated recognition of abnormal structures in wce images based on texture most discriminative descriptors. In: Image Processing and Communications Challenges 2. Springer, pp 263–270
31. Takahashi T, Kudo M, Nakamura A (2011) Construction of convex hull classifiers in high dimensions. Pattern Recogn Lett 32(16):2224–2230
32. Wang Y, Shen XJ, Chen HP (2016) Video face recognition based on the convex hull model of kernel subspace sample selection. Journal of Computational & Theoretical Nanoscience
33. Wang D, Song H, Tie Z, Zhang W, He D (2016) Recognition and localization of occluded apples using k-means clustering algorithm and convex hull theory: a comparison. Multimed Tools Appl 75(6):3177–3198
34. Wang J, Wang Y, Deng C, Wang S, Zhu H (2017) Convex hull for visual tracking with emd. In: International Conference on Audio, Language and Image Processing, pp 433–437
35. Xu Y, Hou W (2017) Calculation of operational domain of virtual maintenance based on convex hull algorithm. In: 2017 Second International Conference on Reliability Systems Engineering (ICRSE), pp 1–8
36. You X, Du L, Cheung YM, Chen Q (2010) A blind watermarking scheme using new nontensor product wavelet filter banks. IEEE Trans Image Process Publ IEEE Signal Process Soc 19(12):3271–84
37. You X, Li Q, Tao D, Ou W, Gong M (2014) Local metric learning for exemplar-based object detection. IEEE Trans Circ Syst Video Technol 24(8):1265–1276
38. Zhang D, You X, Wang P, Yanushkevich SN, Tang Y (2009) Facial biometrics using nontensor product wavelet and 2d discriminant techniques. Int J Pattern Recogn Artif Intell 23(03):521–543

Runzong Liu received the B.S. degree in project management from Nanjing University of Technology, Nanjing, China, in 2004. In 2005, he began to study in the College of Computer Science, Chongqing University, Chongqing, China. He received the Ph.D. degree with the College of Computer Science, Chongqing University, in 2012. He is currently a lecturer with the College of Computer Science, Chongqing University, Chongqing, China. His research interests include machine learning, pattern recognition, computer vision and cognitive science.

Yuan Yan Tang (F'04) graduated from electrical and computer engineering at Chongqing University, China, and received the MEng degree in electrical engineering from Beijing Institute of Post and Telecommunications, China, and the PhD degree in computer science from Concordia University, Montreal, Canada. Currently, he is working as a Chair Professor in Faculty of Science and Technology at University of Macau and Professor/ Adjunct Professor/ Honorary Professor at several institutes including several Universities in China Concordia University in Canada, and Hong Kong Baptist University in Hong Kong. His current interests include wavelet theory and applications, pattern recognition, image processing, document processing, artificial intelligence, Chinese computing. He has published more than 400 technical papers and is the author/coauthor of over 25 monographs/books/bookchapters on subjects ranging from electrical engineering to computer science. He is the Founder and Editor-in-Chief of International Journal on Wavelets, Multiresolution, and Information Processing IJWMIP (SCI), and Associate Editors of several international journals related to Pattern Recognition and Artificial Intelligence IJPRAI (SCI). He is the Founder and Chair of pattern recognition committee in IEEE SMC. He has serviced as general chair, program chair, and committee member for many international conferences including the General Chair of the 18th International Conference on Pattern Recognition (ICPR'06). Dr. Tang is the Founder and General Chair of the series International Conferences on Wavelets Analysis and Pattern Recognition (ICWAPRs). He is a Fellow of IEEE, and Fellow of IAPR (International Associate of Pattern Recognition).

Patrick P. K. Chan received a doctor's degree in the Department of electronic computing from Hong Kong Polytechnic University in 2009. His research focuses on machine learning and pattern recognition, and the research topics include multiple classifiers, antagonistic learning and network security.