A Parallel Dynamic Convex Hull Algorithm Based on the Macro to Micro Model

Haifeng Wan[1], Zhizhuo Zhang[2], Ruijie Liu[1]
1) South China University of Technology, Guangzhou, China
2) National University of Singapore, Singapore
E-mail: wanhaifeng09@gmail.com

Abstract.

In this paper, we analyze how humans can solve problems quickly by the process of solving the problem of convex hull. Then we present the M2M (Macro to Micro) data structure which maintains a finite set of n points in the plane under insertion and deletion of points in amortized O(1) time per operation and O(n) space usage. In addition, as the insert operation of each point is independent, the algorithm has high parallelism. And because the insert operations will not cause the imbalance of the tree structure, the M2M data structure is dynamic. Moreover, it can be shared by all the algorithms based on M2M model which greatly improves the efficiency when a variety of algorithms work simultaneity, such as in image processing and pattern recognition. In all, the M2M model points out a general pattern for designing high parallel algorithms, an efficient strategy for solving multi-operational problems and a new approach for computer to stimulate the thinking pattern of human beings.

1. Introduction

The convex hull of a set of points is one of the most basal problem in computational geometry and is applied in many fields, such as pattern recognition, image processing, statistics and GIS. It also serves as a tool and a building block for a number of other computational geometric algorithms.

Ron Graham presented the first algorithm to compute the convex hull of points in the plane with $O(n\log n)$ complexity in 1972 [1]. If the points are already sorted by one of the coordinates or by an angle to a fixed vector, then the algorithm takes $O(n)$ time. Another solution with $O(n\log n)$ complexity is the divide and conquer algorithm for the convex hull, published in 1977 by Preparata and Hong [2]. This algorithm is also applicable to three dimensions. Later, Avis [4] and Yao [5] proved lower bounds of $\Omega(n\log n)$ to find a convex hull.

R. A. Jarvis constructed an "output sensitive" algorithm whose running time depends on the output size [3]. Jarvis's algorithm runs in $O(nh)$ time where h is the number of vertices of the convex hull. In 1986, Kirkpatrick and Seidel [6] computed the convex hull of a set of n points in the plane in $O(n\log h)$ time. (Later, the same result was obtained by Chan using a much simpler algorithm [7].) They showed that, on algebraic decision trees of any fixed order, $O(n\log h)$ was a lower bound for computing convex hulls of sets of n points, where h was the number of vertices of the convex hull.

Started with seminal work by Clarkson, randomized algorithms have played an increasingly important role in computational geometry and many randomized convex hull algorithms were proposed [8, 9, 10].

Dynamic convex hull algorithms are widely studied [11-17], for we often require computing the convex hull of the point set in practice which is changing in a small scale.

In order to reduce the time complexity, some researchers focused on the techniques of designing fast parallel algorithms for convex hulls [18].

In this paper, we introduce a new efficient convex hull algorithm with dynamic data structure based on M2M model The M2M convex hull algorithm (M2M-CH) is one of the M2M algorithm series. Its dynamic data structure is based on M2M model which was proposed [19]. The preprocessing of M2M-CH takes $O(n)$ time to construct the data structure and supports parallel computation. Based on this data structure, the query procedure shrinks the search space from coarse-grained level to fine-grained level and finally obtains the convex hull in a considerably small search space. Additionally, the M2M model has many desirable properties, such as dynamic structure, preprocessing sharing, trade-off between time cost and accuracy, etc. With these properties, M2M-CH can be applied to various applications and solve problems efficiently.

2. The Outline of the M2M Model

When used for solving problems, the M2M model will work in a macro perspective to shrink the search space until it is sufficiently small, and then focus on a smaller scale of the problem.

Generally, an M2M algorithm includes following two procedures:

Preprocessing

The preprocessing procedure constructs the hierarchical data structure of M2M algorithm from fine-grained level to coarse-grained level. In each level, data set will be divided into a number of similar partitions.

Query

The query procedure shrinks the search space at each level, from coarse-grained level to fine-grained level. Finally, the solution can be obtained at the finest-grained level quickly.

3. Terminology

3.1 Terminology of the M2M Model

Level

The hierarchical data structure consists of levels of different granularity. A coarser-grained level presents abstracted data classification in macro view, while a finer-grained level presents detailed data classification in micro view.

Part

Part is defined as a subset of the data points. At each level, the original point set is divided into squares of the same size. All the data points in such a square belong to this part.

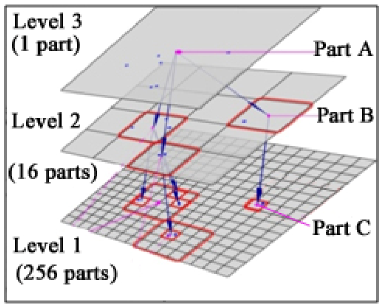


Figure 1. Terminology Explanation

3.2 Terminology of M2M-CH

Center Hull

The convex hull of all the center points of parts that contain at least one point.

Representative Point

An arbitrarily designated point in a part whose center point is the vertex of the center hull.

Representative Hull

The convex hull of all the representative points.

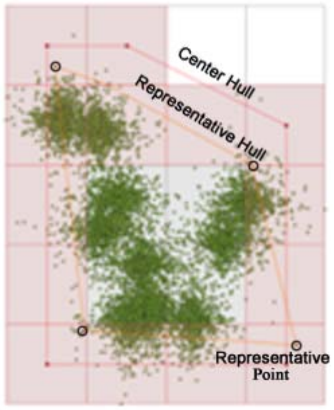


Figure 2. Terminology Explanation

4. The Convex Hull Algorithm Based on the M2M Model

There are two procedures in M2M-CH: preprocessing and querying. Preprocessing constructs the hierarchical data structure. Assume the original point set is at level 1. At level *k*, the graph is divided into square parts of the same size: $Part_1(k)$, $Part_2(k)$, ..., $Part_n(k)$. Each part corresponds to one subset of the original point set. All the points in $Part_i(k)$ are abstracted to the center point of $Part_j(k+1)$ when $Part_i(k)$ is the child part of $Part_j(k+1)$ which is in the next coarser-grained level (Figure 1). If all the child parts of $Part_i(k)$ don't contain any point, $Part_i(k)$ doesn't either. A preset parameter – *Density of* $Part_i(1)$ denotes the maximum number of points in a part at in the level 1. It determines the total number of levels of the hierarchy because the total number of points in $Part_i(1)$ cannot exceed *Density of* $Part_i(1)$. For example, if the number of original points is 12 and the *Density of* $Part_i(1)$ is 3, the number of parts at level 1 is 4. But the number of original points is 13 and the *Density of* $Part_i(1)$ is also 3, the number of parts at level 1 will be 9 because 13/4>3.

```
M2M-CH
Input:
V   // the original point set
M   // M2M data structure built on the original point set
Output : the convex hull of V
M2M-CH(V, M)
1. S← V  //S is the considering point set
2. i←1  //current exploring level
3. While(i≠M.Depth )
4.   do  C← Extract the center points from S at level i
5.       P← QuickHull(C)   //compute the center hull
6.       R← Extract the representative hull from P
7.       S← the point set belonging to the parts which
         intersect with R
8.       i←i+1
9. Return QuickHull(S)
```

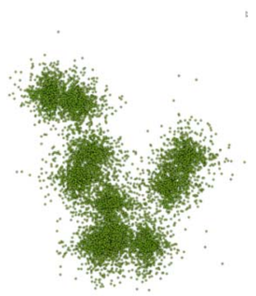
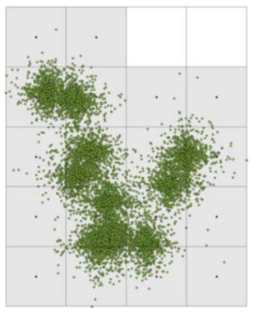
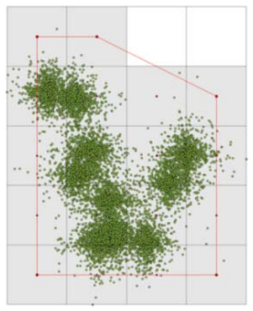
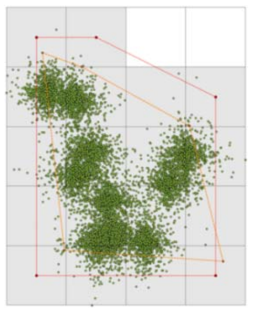
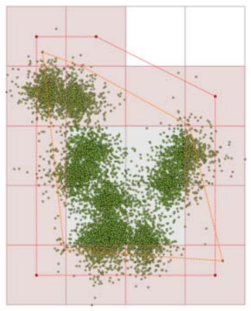
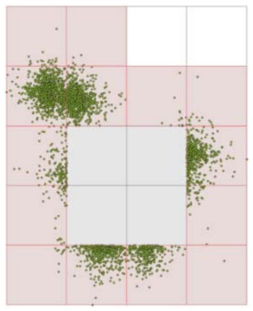
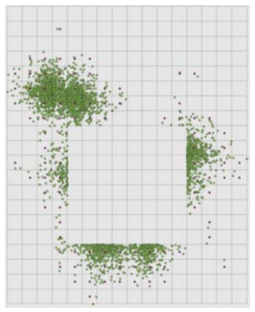
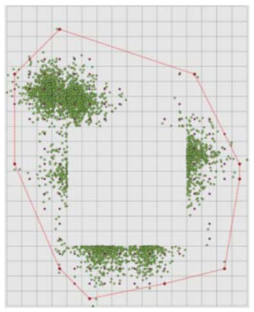
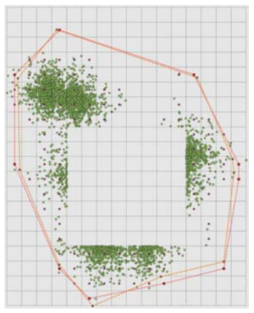
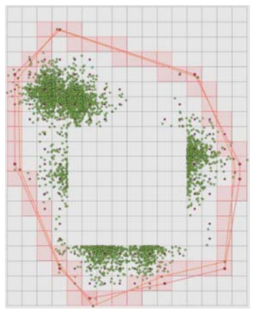
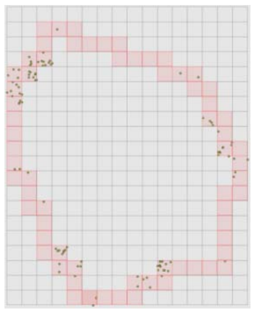
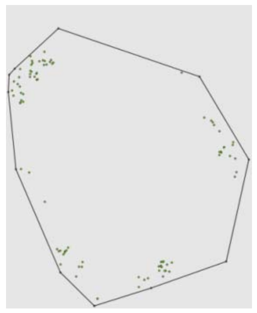
After preprocessing the original point set (Table 1(a)), M2M-CH works as the procedure shown in Table 1. At each level, M2M-CH preserves parts which intersect with the *representative hull* as the search space for the next finer-grained level. Repeat this step until it reaches the finest-grained level. Then M2M-CH computes the resulting convex hull in a considerably small search space. The correctness of M2M-CH will be proved in the next section. An example of the query procedure is shown in Table 1.

Since M2M-CH shrinks search space based on large parts at coarser-grained level, it can quickly exclude a large region which is obviously in the convex hull. Moreover, it is necessary to go through finer-grained level, for the parts in this level are too large for precise operation.

At finer-grained level, M2M-CH shrinks search space based on small part so that it can precisely exclude points which are not the vertices of the convex hull.

In summary, M2M-CH emphasizes on fast process at coarse-grained level, while on accurate solution at fine-grained level.

Table 1: M2M-CH procedure

 (a) The original point set	 (b) The center points in the top level	 (c) The *center hull* of the top level	 (d) The *representative hull* of the top level
 (e) The parts that intersect with the *representative hull* in the top level	 (f) The search space preserved for the second level	 (g) The center points in the second level	 (h) The *center hull* of the second level
 (i) The *representative hull* of the second level	 (j) The parts that intersect with the *representative hull* in the second level	 (k) The search space preserved for the bottom level	 (l) the final convex hull

5. Proof of the Correctness of M2M-CH

To prove the correctness of M2M-CH, we introduce the following lemmas first:

Lemma 1

At any level, a part whose center point is outside of the *center hull* contains no point.

Proof

Considering a part at level k, namely $Part_i(k)$, whose center point is p_i. If $Part_i(k)$ contains a point and p_i is outside of the *center hull*, then it contradicts to the definition of convex hull that requires all the points, including p_i, should be inside the convex hull. Hence, $Part_i(k)$ contains no point. Lemma 1 is proved.

Lemma2

At any level, a part which is completely inside of the *representative hull* contains no hull point.

Proof

Considering a part at level k, namely $Part_i(k)$, which has a point p_i. If p_i is a hull point and $Part_i(k)$ is completely inside of the *representative hull*, then there must be at least a point q_i outside of the resulting convex hull, where q_i is the vertex of the *representative hull*. It contradicts to the definition of convex hull. Hence, $Part_i(k)$ contains no hull point. Lemma 2 is proved.

Lemma 3

At any level, a part whose center point is inside of the *center hull* and outside of the *representative hull* has an intersection with the *representative hull*.

Proof

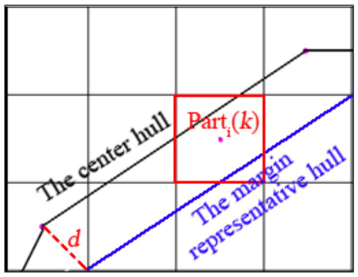


Figure 3. Proof of Lemma 3

Considering a part at level k, namely $Part_i(k)$, whose center point is p_i. p_i is inside of the *center hull* and outside of the *representative hull* (Figure 3). We define the margin *representative hull* is a *representative hull* with the smallest size and the side length of $Part_i(k)$ is 1. Let d denote the distance between the corresponding lines of the *center hull* and the *representative hull.* Figure 3 clearly shows that the maximum value of d is $\frac{\sqrt{2}}{2}$, smaller than the side length of $Part_i(k)$. Hence, $Part_i(k)$ must have an intersection with the *representative hull*. Lemma 3 is proved.

Lemma 4

At any level, a part which contains hull points has an intersection with the *representative hull*.

Proof

Considering a part at level k, namely $Part_i(k)$, whose center point is p_i. If $Part_i(k)$ contains hull points and has no intersection with the *representative hull*, then it must be either completely inside or outside of the *representative hull*. If it is completely inside of the *representative hull*, according to Lemma 2, it contains no hull point. If it is completely outside of the *representative hull*, then p_i is outside of the *center hull*. According to Lemma 1, it contains no hull point. Then, it is concluded that $Part_i(k)$ has no hull point if it has no intersection with the *representative hull*, which contradicts with the hypothesis. Hence, Lemma 4 is proved.

Now, we prove the correctness of M2M-CH using the following loop invariant.

Proof

Initialization

The querying processing begins from the coarsest-grained level. The search space in this level includes all the points of the original points set. Hence, at the initialization, all hull points are included.

Maintenance

M2M-CH preserves all the parts which have intersection with the *representative hull* while exclude other parts from the search space. According to lemma 4, the search space preserved for the next finer-grained level contains all the hull points. It guarantees that no hull point is excluded while shrinking the search space.

Termination

Since all hull points are in the search space of the finest-grained level according to loop invariant in maintenance, the inner algorithm can generate the correct convex hull. This completes the proof.

6. Experiment

In the following experiments, we use some classical convex hull algorithms (Graham scan [1], quick hull [2] and Jarvis march [3]) as benchmark algorithms to analyze the performance of M2M-CH. We analyze the efficiency of these algorithms by the same point set.

Figure 4 shows that when computing the convex hull of a point set whose scale is small, Graham's scan has the best performance. As scale of the point set increases, M2M-CH outperforms other classical algorithms. Figure 5 shows the relationship between the time cost of the query procedure of M2M-CH and the scale of the point set.

The time complexity of the preprocessing of M2M-CH is O(n), for the preprocessing consists of n times of insertion operations and the amortized time cost of each insertion operation is O(1). But the time complexity of the query procedure is hard to estimate, for it depends on the distribution of the point set. The worst situation is that all the points are the vertices of the convex hull. In this situation, the time complexity of M2M-CH is O(nlogn), the same as those traditional algorithms. But this pathological ordering doesn't occur in practice. In most cases, M2M-CH excludes a large amount of points in the convex hull level by level. This greatly reduces the problem scale. Thus, the time cost of the query procedure is much smaller than that of the preprocessing.

Table 2: Abbreviation Denotation

M2M-CH	The M2M convex hull algorithm
GSCH	Graham scan algorithm.
QHCH	Quick hull algorithm.
JMCH	Jarvis's algorithm

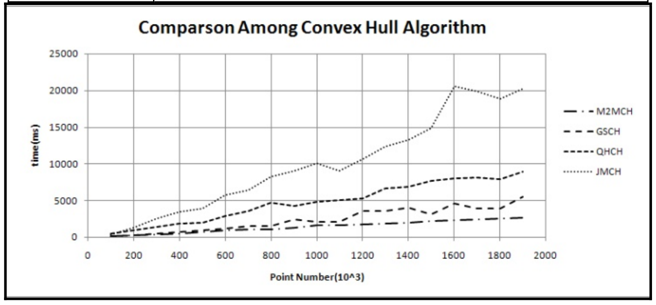


Figure 4. Comparison among convex hull algorithms

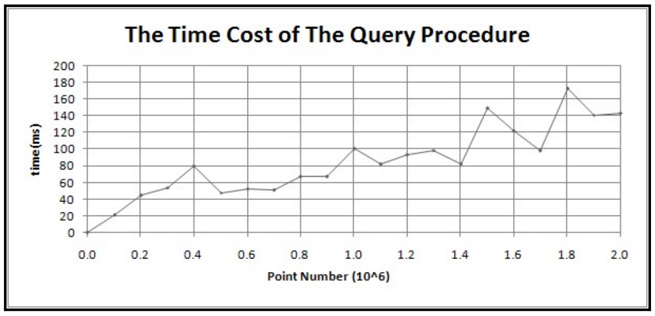


Figure 5. The time cost of the query procedure

7. The Advantages of Convex Hull Algorithm of the M2M Model

Compared to conventional convex hull algorithms, M2M-CH has following advantages:

High parallelism

Although the preprocessing of M2M-CH occupies a great proportion of the total time, it can be run by parallel computing device, for operations on different points are independent. Hence, M2M-CH has great potential to reduce the overall time cost.

Dynamic structure

The operation of M2M data structure such as insertion or deletion can be finished in $O(1)$ time in most cases while in $O(\log n)$ time in the worst case. Hence, there's no need to reconstruct the hierarchical structure when the original point set changes a little, but update the information of those new points and the parts they belong to.

Preprocessing sharing

Preprocessing sharing is very helpful in the image processing field where many operations need to be executed on the same image. For instance, in face recognition, we probably require computing the convex hull in different regions of the same graph. In this case, M2M-CH only constructs the hierarchical structure once, and then it can compute the convex hull of different regions quickly according to preprocessing sharing.

Trade-off between time cost and accuracy

M2M-CH can make a trade-off between the efficiency and the accuracy of the solution by outputting the *representative hull* of level k (k>1) instead of the convex hull of the finest-grained level. In this case, the time cost is shorter while the result is just approximately correct. It is obvious that the smaller the value of k we set, the more precise the resulting convex hull is. In this way, M2M-CH can be applied to various applications with different demands on time cost and accuracy.

Extension to Three Dimensions

Since the M2M model is based on regular parts, M2M-CH can be extended to three dimensions by defining a part as a cube.

8. Conclusion

From the experiment result, we can conclude that the efficiency of M2M-CH is better than that of classical algorithms when the scale of the point set is comparatively large. More importantly, the M2M data structure is dynamic and can be constructed in parallel. The amortized time complexity of its update operations is optimal.

Based on the M2M model, M2M-CH has many desirable characteristics, such as high parallelism, dynamic, preprocessing sharing, trade-off, etc. These characteristics play a significant role in various applications. The M2M model introduces a general model to design algorithms with high parallelism, an efficient strategy to handle multi-operational problems and a new approach to stimulate the thinking pattern of human beings. With the help of the M2M model, computer can solve problems with a wider holistic view like human.

References

[1] R. L. Graham, An efficient algorithm for determining the convex hull of a finite planar set, Information Processing Letters 1 (1972) 132–133.

[2] F. P. Preparata, S. J. Hong, Convex hulls of finite point sets in two and three dimensions, Communications of the ACM 2 (20) (1977) 87–93.

[3] A. Jarvis, On the identification of the convex hull of a finite set of points in the plane, Information Processing Letters 2 (1973) 18–21.

[4]Avis, D. Comments on a lower bound for convex hull determination. Inform. Process. Lett.11 (1980), 126.

[5]Yao, A.C. A lower bound to finding convex hulls. J. ACM 28 (1981), 780-787

[6] D. G. Kirkpatrick, R. Seidel, The ultimate planar convex hull algorithm, SIAM Journal on Computing 15 (1) (1986) 287–299.

[7] T. M. Chan. Optimal output-sensitive convex hull algorithms in two and three dimensions. Discrete Comput. Geom., 1996. Eleventh Annual Symposium on Computational Geometry.

[8]Clarkson, K.L. New applications of random sampling in computational geometry. Discrete Comput. Geom. 2(1987), 195-222.

[9]Clarkson, K.L. A randomized algorithm for closest-point queries. SIAM J. Comput. 17(1988), 830-847.

[10] R. Wenger, Randomized quick hull, Algorithmica 17 (1997) 322–329.

[11] M. H. Overmars and J. van Leeuwen. Maintenance of configurations in the plane. J. Comput. System Sci., 23(2):166–204, 1981.

[12] J. Hershberger and S. Suri. Applications of a semi-dynamic convex hull algorithm. BIT, 32(2):249–267, 1992.

[13] J. Hershberger and S. Suri. Off-line maintenance of planar configurations. J. Algorithms, 21(3):453–475, 1996.

[14] T. M. Chan. Dynamic planar convex hull operations in nearlogarithmic amortized time. Journal of the ACM, 48(1):1–12, January 2001.

[15] Gerth Stolting Brodal and Riko Jacob Dynamic Planar Convex Hull. 43 rd Annual IEEE, 2002.

[16] Maarten Löffler and Marc van Kreveld, Largest and Smallest Convex Hulls for Imprecise Points, OpenAccess, 2008.

[17] Rong Liu, Hao Zhang and James Busby, Convex Hull Covering of Polygonal Scenes for Accurate Collision Detection in Games, Graphics Interface,2008

[18] Neelima Gupta and Sandeep Sen, Faster output-sensitive parallel algorithms for 3D convex hulls and vector maxima, Journal of Parallel and Distributed Computation, 2003.

[19] YingPeng Zhang, ZhiZhuo Zhang, Qiong Chen A NEW NEAREST NEIGHBOUR SEARCHING ALGORITHM BASED ON M2M MODEL. THE INTERNATIONAL MULTICONFERENCEOF ENGINEERS AND COMPUTER SCIENTISTS 2007, 2007.