A New Nearest Neighbour Searching Algorithm based on M2M Model

YingPeng Zhang, ZhiZhuo Zhang, and Qiong Chen

Abstract— **In this paper, we introduce a generally applicable algorithm model named Macro-to-Micro Model (M 2M) which is derived from human thinking pattern. The data structure for the nearest neighbor problem based on M 2M can be built in O(n) time. It can also be finished in O(1) time by parallel technology. Moreover, the insertion, deletion and query operation can be completed in constant time without the problem of breaking the balance of tree. And the most noteworthy is that this data structure and preprocessing operation can be shared with most M2M algorithm, so that we can hugely improve the efficiency of the multi-operation problem like image processing and pattern recognition. We mainly focus on the nearest neighbour (NN) searching algorithm in this paper. The M2M approach can achieve the optimal expected time complexity. And the comparative experiment between M 2M and kd-tree shows the great advantage of the former.**

***Index Terms*— Macro-to-Micro (M2M), nearest neighbour searching, closest point problem.**

I. INTRODUCTION

Nearest neighbor searching is the following problem: Given a set of n data points in a metric space (this paper focus on the problem in the planar space) and the problem is: which point is the nearest point to the query point in this data set. This is also called the closest-point problem. Nearest neighbor (NN) algorithm is a fundamental algorithm in many application field including knowledge discovery and data mining (Fayyad et al. 1996) pattern recognition and classification (Cover and Hart 1967; Duda and Hart 1973), machine learning (Cost and Salzberg 1993）, data compression(Gersho and Gray 1991), multimedia databases (Flickner et al. 1995), document retrieval(Deerwester et al. 1990).

Account for the great influence of NN algorithm, many researchers are keeping study on it. The seminal paper in this field is the classic work of Shamos and Hoey [1] in which the problems are defined and a number of optimal worst case algorithms for planar point sets are given. Randomized algorithms for the closest pair problem have been given by Rabin [2] and Weide [3]; Fortune and Hopcroft [4] have shown that the speedup of the fast closest pair algorithms was not due to their randomized nature alone, but also to the model of computation employed (which allowed floor functions). More theoretical results were generalized by Bentley et al[5] who analyzed a grid-base method for distributions satisfying certain bounded-density assumptions and showed that O(n) preprocess time and constant query time in the expected case.

The approximate algorithm based on quadtrees (Bern et al [6][7]) uses linear space and provides logarithmic query time. O(n) space and O(log n) query time are achievable in the expected case in the use of kd-trees proposed by Friedman et al[8]. These results were generalized by ARYA et al [9], who proposed a data structure called BBD-tree and design an optimal algorithm for approximate nearest neighbor searching in fixed dimensions.

Our comparison of kd-tree follows an explanation given by Andrew Moore in his PhD thesis [10], who, along with Omohundro (1987), pioneered its use in machine learning. Moore describes sophisticated ways of constructing ball trees that perform well even with thousands of attributes[11].

In this paper, we propose a generally applicable algorithm model which is named Macro to Micro Model (M2M), and develop an efficient NN searching algorithm with the optimal expected time based on M2M model .The data structure based on M2M model (the M2M structure) only cost a linear time to preprocess the data and the other tree structure such as kd-tree, costs O(nlogn) time. By using the parallel computation, the time complexity can be reduced to O(1). The operations including querying, insertion and deletion on other conventional structure (like kd-tree and quadtree) take O(logn) time, which may also cause balance problem of the tree. However, those operations are independent on each point in the M2M structure and take constant time. A more distinguished trait of M2M model is that the preprocessing can be shared by the other algorithms based on M2M model, which greatly improve the efficiency of some multi-operation problem like image processing and pattern recognition.

II. MACRO-TO-MICRO MODEL

A. The origin of M2M model

The idea of M2M is derived from human thinking pattern. When people tackle practical problems, they used to analyze at macro level at first rather than details. Then go on specifying the problem's scale more narrowly in order to exclude some unnecessary factors until it reach an appropriate micro level to solve the problem rapidly. With the M2M model, our computer can have the ability as human to comprehend a problem from a

YingPeng Zhang is the student in South China University of Technology, Guangzhou, China 510640 (e-mail: YingPeng.Zhang@gmail.com)

ZhiZhuo Zhang is the student in South China University of Technology, Guangzhou, China 510640 (e-mail: zzz2010@gmail.com)

Qiong Chen is the associate professor in South China University of Technology,Guangzhou, China 510640 (e-mail: csqchen@scut.edu.cn) .

global vision. In the more abstract view, the process from macro to micro is achieving the goal of shrinking the search space. In fact, this idea is inherent in many algorithms of "Decrease-and-Conquer". Using M2M model to solve problems include the following two steps:

1) **Preprocess:** Data set should be divided into a number of similar partitions through Macro to Micro levels. This process is similar to the human developing the view of the problem.
2) **Compute:** From macro to micro**,** shrinking the search space at every level and using the algorithms based on M2M to find the solution quickly.

B. *Terminology Explanation*

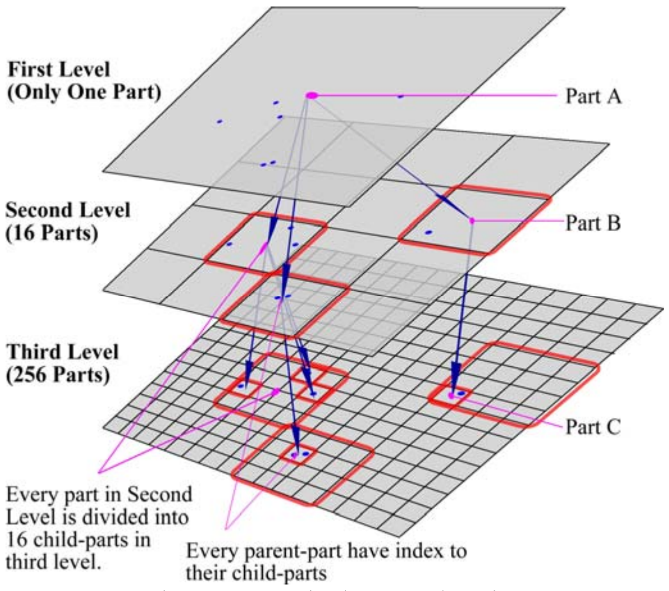

First Level
(Only One Part)
Part A
Second Level
(16 Parts)
Part B
Third Level
(256 Parts)
Part C
Every part in Second Level is divided into 16 child-parts in third level.
Every parent-part have index to their child-parts

Figure 1. Terminology Explanation

Before the further introduction of the data structure based on M2M model, we firstly explain some terminologies which are used frequently when describing the M2M model.

1) **Level:** From the abstract view, different levels present the different way of data classification according to the different precision. As the figure1 shows, there are three levels. Only one part belongs to the first level, 16 ones belong to the second and 256 belong to the third.
2) **Part:** From the abstract view, part is defined as subset of the data points similar to each other. In the M2M model, the part can be designed as a small square. All the data set in the square belong to such partition. In addition, the size of the parts nearly the same in the same level (size is referred to the cover area in the two-dimension).
3) **Last-level, next-level, up-level and down-level:** We define the level according to their parts' size. The size of certain level is smaller than its last-level and the size of certain level is bigger than its next-level as well. Take the figure1 as example, the first level is the last-level of the second level. Then the third level is the next-level of the second level. All the next-levels under the certain level can call the down-level and all the last-levels upper the certain level can call the up-level as well.
4) **Parent-part, child-part, ancestor-part and descendant-part:** The parent-part of a certain part refers to the part belonged to the last-level and contain this part. Similarly, the child part of a certain part is defined as the part included by this part in the last-level. Just as the figure1, part A is the parent-part of part B, part C is the child-part of part B. All the parent-part belonged to certain part can call the ancestor-part of this part. Similarly, all the child-part belonged to certain part can call the descendant-part of this part.

III. THE NEAREST NEIGHBOR ALGORITHM BASED ON M2M MODEL

A. *The data structure based on the M2M model*

M2M is an algorithm model. Many algorithms can be approached based on this model. The data structure to realize this model is flexible, basing on different situations. However, there are some basic requirements needed to be satisfied.

1) Given a query point, it should take O(1) time to index the part which the point belongs to of the given level.
2) Insertion or deletion of a data point should only take O(1) time.
3) Given the index of the part, it takes O(n) time to visit every child-part of the given part, where n is the number of the child-part.
4) Given the index of the part, it takes O(n) time to visit all the data points of this part, and n is the number of the data points of this part.
5) Given the index of the part, it takes O(1) time to get the index of its ancestor-part.
6) The time complexity of preprocessing should be O(n). And support parallel calculation.

It is clear that the algorithm which satisfies all the requirements above achieve the trivial lower bounds and is theoretical optimal.

In order to satisfy those requirements, the following data structure is used in our nearest neighbour searching algorithm in the planar case:

1) A 2-dimension array index is applicable for every part in the same level. Because querying, inserting or deleting an array element only cost O(1) time, the 1st and 2th requirements are satisfied.
2) Every part maintains the index list of its child-parts， so that the 3th requirement can be satisfied through visiting the child-parts list (alternately, when the number of child-parts is small, the space of the index list is no longer needed because of the regular partition).
3) The most micro part maintains a list of the points it contains. When we want to visit the points in a certain part, the breadth search tree can be built by taking query part as the tree root， then we can traversal every point belong to this part. The time complexity of this process is O(n), that is, the 4th requirement is also satisfied.

4) Because the partition are regular, it is easy to calculate the index of parent-part by the index of current part (accomplished by a multiplication for scaling and a floor function to find the integer part index).This process is finish in constant time, so the 5th requirement is satisfied.
5) Because the preprocess is composed of a series of insertion. As we explain at the 2nd statement above, every insertion cost O(1) time, therefore the time complexity of preprocess is O(n) and also support parallel computation which satisfies the 6th requirement.

In this section, we discuss whether 6 requirements above are still satisfied when hash table is used for storage instead of array. We use hash table to maintain a 2-dimension index and use the Channing strategy for the conflict resolution. And we give the analysis below:

Let n denote to the number of the unit in the length of level. So there are n^2 parts in this level at most. We set up a hash table maintain a 2-dimension array which size is m^2 (m < n). We define the load factor α for this hash table as n^2/m^2. We assume that the element being searched for is equally likely to be any of the n^2 elements stored in the table. The number of elements examined during a successful search for an element p which index is (x, y) is 1 more than the number of elements that appear before p in its list. Elements before p in the list were all inserted after p was inserted. So the expected number of elements examined is 1 plus the expected number of elements added to the list after p was added. Let pi denote the ith element inserted into the table, for i = 1, 2, ... , n, and let ki = key[pi]. For keys ki and kj , we define the indicator random variable Xij = I{h(ki) = h(kj)}. Under the assumption of simple uniform hashing, we have Pr{h(ki) = h(kj)} = $1/m^2$,So E[Xij] = $1/m^2$. Thus, the expected number of elements examined in a successful search is

$$
\begin{aligned}
& E\left[\frac{1}{n^2}\sum_{i=1}^{n^2}\left(1+\sum_{j=i+1}^{n^2}X_{ij}\right)\right] \\
&= \frac{1}{n^2}\sum_{i=1}^{n^2}\left(1+\sum_{j=i+1}^{n^2}E\left[X_{ij}\right]\right) \\
&= \frac{1}{n^2}\sum_{i=1}^{n^2}\left(1+\sum_{j=i+1}^{n^2}\frac{1}{m^2}\right) \\
&= 1+\frac{1}{n^2m^2}\sum_{i=1}^{n^2}(n-i) \\
&= 1+\frac{1}{n^2m^2}\left(\sum_{i=1}^{n^2}n^2-\sum_{i=1}^{n^2}i\right) \\
&= 1+\frac{1}{n^2m^2}\left(n^4-\frac{n^2(n^2+1)}{2}\right) \\
&= 1+\frac{n^2-1}{2m^2} \\
&= 1+\alpha/2-\alpha/2n^2 \qquad (1)
\end{aligned}
$$

Thus, the total time required for a successful search (including the time for computing the hash function) is O(2 + α/2 - α/2n) = O(1 + α).

If the number of hash-table slots is proportional to the number of elements in the table, we have n^2 = O(m^2) and, consequently, α = n^2/m^2 = O(m^2)/m^2 = O(1). Thus, searching takes constant time on average. Since insertion takes O(1) worst-case time and deletion takes O(1) worst-case time when the lists are doubly linked, all dictionary operations can be supported in O(1) time on average [12].

It satisfies the basic requirements of M2M's data structure as well. Further more, the space complexity only depending on the scale of points set rather than the number of part. Thus, the strategy we usually used is:
1) Array is used in more macro level considering its fast random accessing.
2) Hash table is used in more micro level for saving the storage cost.

In addition, the number of levels of M2M is defined to log(n) here, but the experiments show that it increases so slowly with the n growing. The number of levels in the case of 100000 points only equals to 5. Thus, we take it as a constant in this analysis. Then the space complexity of M2M structure is presented by its most micro level. As we say above, it is O(n) with the hash table. The space complexity is the same to other traditional data structure such as kd-tree and the quadtree. Empirically, the space cost of M2M structure is a bit more than the traditional ones when it achieve the optimal efficiency.

Because there isn't any dependency of each data points in the M2M preprocessing, it is convenient to run the preprocess in parallel, which will improve the efficiency greatly. And we give the proof below:

let s denote the fraction of total execution time spent in serial code, t is the real time spent in serial code .according to **Gustafson-Barsis's law**, The maximum speedup ψ achievable when p equals the number of points n by this program is:

$$s=\frac{\sigma(n)}{\sigma(n)+\varphi(n)/p}=\frac{t}{t+\varphi(n)/n}$$

$$
\begin{aligned}
\psi &= p+(1-p)s \\
&= p+(1-p)\frac{t}{t+\varphi(n)/p} \\
&= n+(1-n)\frac{t}{t+\varphi(n)/n} \qquad (2)
\end{aligned}
$$

Using the parallel technology, the time complexity of preprocess can be reduced to

$$O(n)/\psi=\frac{O(n)}{n+(1-n)\frac{t}{t+\varphi(n)/n}}=O(1)$$

(Where t is constant)

It means that the time complexity of preprocessing is O(1) by using O(n) processor units. According to parallelism folding principle, it is so convenient to make full use of the existing computation resource and the time complexity of the algorithm changes from O(1) to O(n) correspondingly.

It seems that the M2M structure is similar to other data structure, and we will do a further comparison among them in section V.

B. The approach of nearest neighbour searching algorithm based on M2M

The NN algorithm based on M2M has two steps just as other NN searching algorithm.

1) **Preprocess:** this process is identical to the common M2M preprocess mentioned in section A. Traversal all the data points and make their index in the most micro level. Then build up the parts from micro level to macro level indexing to their parent parts.
2) **Solution:** As M2M is a generally applicable algorithm model, there is not just one NN algorithm can run on the same M2M structure. Here, we just present the optimal one in our experiment. This M2M algorithm has two process as well:
    a) **Level Selection:** The goal of this process is to find appropriate level for searching. It starts at the most macro level ,and go downward to the part contain the query point (we call it the query part).If the query part contain other points except query point ,continue searching the child-part in the next-level until the child-part is empty or reach the most micro part. Then enter into the second process.
    b) **Searching Process:** After the level selection, the searching level has been determined. Next, traversal all the points in the query part of selected level. A current nearest point can be found, if the part isn't empty, Else, we search the parts surrounding the query one in an expanding pattern until a point is found. Once we have one point, we are guaranteed that there is no need to search any part without intersection with the circle of radius equal to the distant to the current nearest point and centered at the query point. In order to make it easy to understand, we use a square instead of the circle in Figures below. Through searching the parts intersecting the square, a new nearest point may appear and the square becomes smaller, Finally, all the parts intersecting the square have been searched ,and at that time ,the current nearest point is exactly the nearest neighbour.

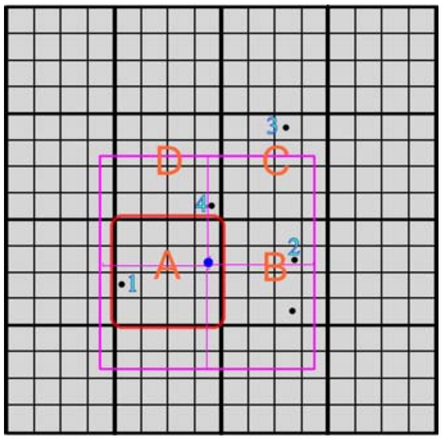


Figure 2
Querying part A, current nearest point 1 is found.

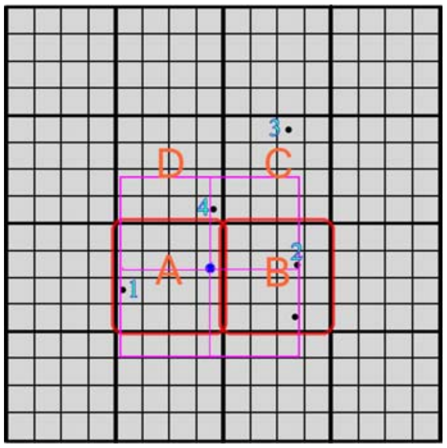


Figure 3
Searching bound shrinked, found current nearest point 2 in part B

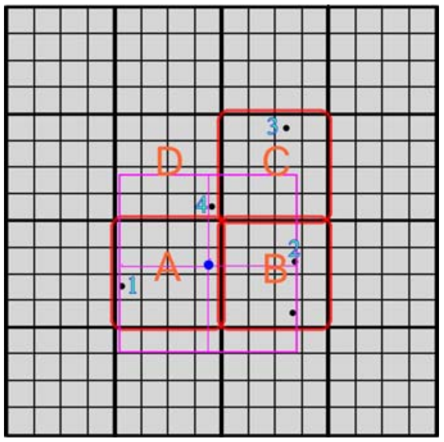


Figure 4
Querying Part C, Searching bound no change

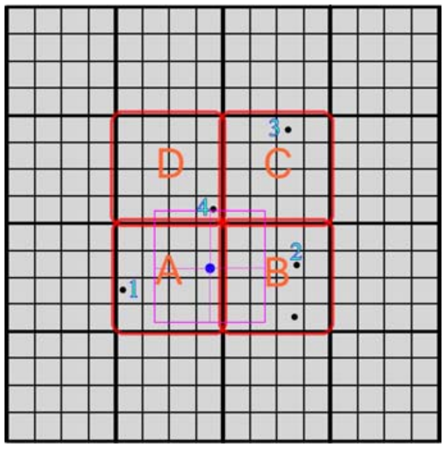


Figure 5
Searching bound shrinked again and all area intersected has been searched, found point 5 is global nearest point

The best case of our algorithm is that the searching circle is included by the query part after searching that part. By contrast, we also can find a worse case in which the algorithm should compare with all the points in case that all the point distributes on a circle centered at the query point. But that is exactly the worse case of kd-tree and quadtree as well.

Calculation of the expected time for NN is difficult, because the analysis depends critically on the level, and the expected distribution of the data points presented to the nearest neighbour algorithm. The analysis for kd-tree is performed in log(n). Research shows that the expected number of intersecting hyperrectangles is independent of N, the number of exemplars [Moore et al. 1991; Friedman et al. 1977]. And M2M -NN algorithm has the similar propriety. Bentley et al have proofed that searching process takes O (1) expected time in uniform distribution in their paper [5]. And we only give an explanation here.

Let's denote P is the query point and S is the collection of the searched parts of the top level. As the algorithm works, a number of points can be inserted into the part r ($r \notin S$) without any influence on the process of searching the nearest neighbour of P; In other words ,the time complexity of the algorithm is independent of the number of the point set . And we show this conclusion in the experiments below.

C. *Empirical Behaviour of M2M-NN*

To learn more about the practical performance of M2M algorithm, we construct the following experiments of comparison to the kd-tree coded by Sebastian Nowozin [13] based on the paper of Andrew W. Moore [10].

In the following experiments, planar points were generated randomly from uniform distribution. The time was calculated by the average time cost in 1000 queries.

1) **Comparison of preprocessing time** :

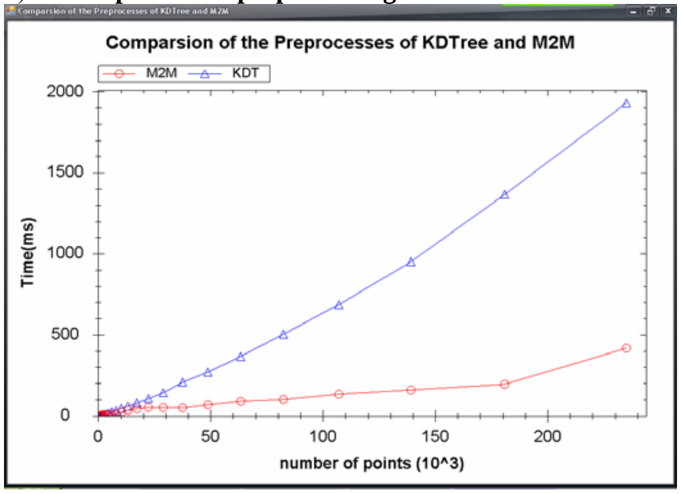


Figure 6.

Table 1. Comparsion of the preprocesses of kd-tree and M2M

number of point set (thousand)	kd-tree	M2M
2	7.227ms	4.395ms
10	46.875ms	23.4ms
50	270.833ms	69.444ms
235	1930.555ms	418.4ms

As figure 6 shows, the experiment result accords with the theoretical analysis. In the preprocess of kd-tree, selecting a good pivot costs O(logn) time at every level[Moore et al. 1991; Friedman et al. 1977]. Therefore, the total time of building tree is O(nlogn) and we can see it increase nonlinearly with the number of points growing . However, the preprocess of M2M only cost O(n) time and it is self-proof by the figure 6.

2) **Comparison of the time of searching nearest neighbour** :

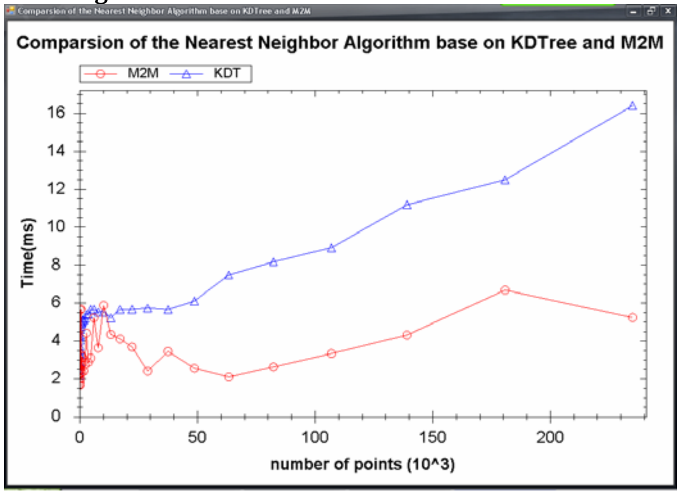


Figure 7.

Table 2. Comparsion of the nearest neighbor algorithm base on kd-tree and M2M

number of point set (thousand)	kd-tree	M2M
2	5.216ms	2.739ms
10	5.537ms	5.885ms
50	6.116ms	2.566ms
235	16.406ms	5.233ms

As we have analyzed before, the searching time of kd-tree is O(logn), M2M approach is approximative to O(1) which also shown in the figure. In addition, there are several undulation with the number of points growing, because of the number of level is changing discontinuously(by floor function).

What is the best way to divide each level, and how many level is sufficiency remain problems. There is no mature theory for it up till now. But we come to some conclusions in the experiment for 10000 data points.

Table 3. Comparsion among different number of part and different number of level.

P\L	3	4	5
9	474.75	67.86	12.2
16	179.448	17.25	3.444
25	77.6	7.03	2.55
36	42.71	3.9814	2.8777
49	25.447	2.784	2.538

(P: The ratio of the size of the adjacent levels, L: The number of level)

Form this table we can catch that along with the increase of the number of level, the number of points which are compared to is converge at some value. Considering time and space cost, 5 is a good choice for the number of part.

The experiments for the performance of insertion and deletion won't be carried out here, because the experiment result of dynamical insertion or deletion is identical to the preprocessing. As we have explained, the preprocessing is just composed of n times independent insertion where n is the number of data points. And deletion is almost the same process as insertion except which is not adding but removing the index of points. Therefore the structure after deletion or insertion has as good performance as after preprocessing.

IV. Discussion

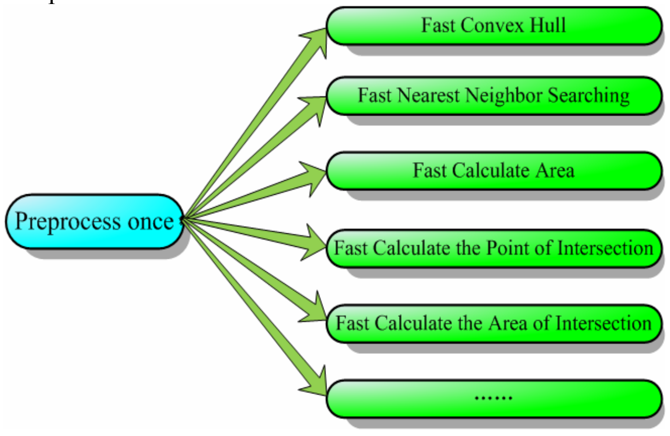
A. The advantage of M2M model

M2M model have following advantages:

1) **High parallelism of preprocessing:** The preprocessing of M2M can be run in multi-independent channels at the same time and solved in constant time with n processor (where n is the number of the data points). This can greatly improve the preprocessing efficiency of M2M in the multi-CPU, multi-kenel or grid computation environment.
2) **Preprocessing sharing:** The algorithms based on M2M model share an identical preprocessing. It is very helpful in the image process field where many operation need to be executed on the same image (point set as well) and many algorithms used in image process can be approached based on the M2M model. All those algorithms share one preprocessing, thus the efficiency of the whole processing is greatly improved.
3) **Shrinking the search space:** From macro level to micro level, M2M algorithm shrinks the searching space at every level. As a result, it can shorten the searching time.
4) **Trade-off between the efficiency and the precision:** In most case, M2M algorithm can trade-off between the efficiency and the precision easily. It's very useful, for sometimes people are willing to get a approximate result in order to reduce the computing time.
5) **Trade-off between time efficiency and space cost:** The parameters of M2M such as the number of levels and the way of partition at each level can be changed on certain purpose. General speaking, the more sufficiently of the Macro-Micro levels being divided and the smaller ratio of the partition between the adjacent levels, the higher cost of the space may be, but the higher efficiency may get.

B. Other application of M2M model

With the help of M2M modeling, computer can be more nature, more flexible and more efficient to solve many classical algorithm problems; and the problems in specific domain (such as nearest neighbor, convex hull, TSP, cluster, path finding, collision detection and so on) can be designed the corresponding M2M algorithm. Thus, tasks in many application fields (including geography information system, data mining, pattern recognition, image processing and real time strategy game) related to those fundamental algorithm can be better performed. Taking image processing for example, we can preprocess for a specific point set, and then does a fast convex hull, a fast nearest neighbor, or a fast area calculation in this point set.


Preprocess once
Fast Convex Hull
Fast Nearest Neighbor Searching
Fast Calculate Area
Fast Calculate the Point of Intersection
Fast Calculate the Area of Intersection
……

V. Related Work

1) M2M and Divide-and-Conquer (DAC)

The similarity between them is both of them shrink the scale of problem through dividing part or separate level technology. On the other side, there is an obvious difference between them. That is, DAC divide the problem into sub problem and solve each of them. But M2M divide the problem into sub-problem and discard the states which no need to concern. Through search in difference level, M2M shrinks the search space, but the problem left is still integral. In a sense, it implements the Decrease-and-Conquer.

2) M2M and Voronoi diagram

The NN algorithm based on Voronoi diagram is faster than the approach based on M2M(why?). But the preprocess time of it is O(nlogn). In fact, maybe we can find a way to build the Voronoi diagram in O(n) time by using M2M approach.

3) M2M and cell technology

Some researchers have proposed the data structure of regular partition called "cell technology" [Bentley et al. 1980; Rivest et al. 1974; Cleary et al. 1979] which can do well in the situation of uniform distribution. But they didn't put the regular partition strategy and hierarchical decomposition of space together. So the disadvantage of the algorithm as follows:

a) The greater clustering led to a greater degradation of performance
b) The Algorithm performs poorly in the non-uniform distribution.

Two problems above are from the same reason: a fixed way of partition can not do well in all the distribution. To the contrary, the M2M-NN, choose a suitable level before searching, so that it has good performance in various distributions.

4) M2M and other data structure based on hierarchical decomposition of space

It seems that the M2M data structure is similar to the other data structure based on a hierarchical decomposition of space, such as balanced box-decomposition(BBD) tree [SUNIL ARYA et al. 1998] , quadtree [Bern et al. 1993], kd-tree

[Friedman et al. 1977] or even ball-tree (sometimes called a metric tree)[Moore et al. 2000; Alexander Gray et al 2001]. (Neighboring spheres may overlap in ball tree whereas rectangles can abut in other tree structure.) But, there are essential differences between M2M structure and other tree structure.

a) The operations like getting the index of a certain part, inserting or deleting a point, only cost O(1) time in the M2M structure, but O(log(n)) in quadtree or kd-tree even ball-tree and BBD-tree.
b) In the M2M structure, every point belongs to all levels, So that observation on a point can be easily made in different level. However, in preprocess of kd-tree, quadtree, every point has been fixed in a certain part of a certain level. As a result, the points can not be view at different levels and that is the reason why there are difficulty to insert and delete from those data structure.
c) In the M2M structure, the scale of every part in the same level is similar and comparable.
d) Although M2M structure is a complicated tree structure as well, it is not necessary to consider the balance problem. In M2M structure, operating on one point won't affect the others and the time cost of insertion and deletion are O(1) as well. However, the balance problem is inevitable in KD-tree and quadtree. As a result, doing an insertion or deletion is so expensive to them [Moore et al. 1991; Friedman et al. 1977].
e) The number of child-part can be configured flexibly according to the consideration of space and time efficiency.

VI. CONCLUSION

In this paper, we present M2M model and the nearest neighbour searching algorithm based on M2M model. The operations of the data structure of M2M model (including preprocessing, insertion, deletion and searching the nearest neighbor) are achieve the optimal expected time complexity. What's more, the independency of the data in the structure of M2M model can avoid the balance problem in other tree structure like kd-tree, BBD-tree or quadtree. And using the parallel technology, the time complexity of preprocessing can be reduced to O(1). Finally, the potential advantage of the M2M structure is the preprocessing can be shared by different algorithms based on M2M model in the same data set.

Much further work remains to be done to optimize NN algorithm based on M2M model.

1) **Select the searching level more precisely:** If the search starts at the macro level, the number of the points being compared may be large. On the other sides, the number of the part being examined may be large in the micro level.
2) **Expand the searching area efficiently:** The better interpretation is circle but it is more complicated to determine whether a part is inside the area. For that reason, we use the square instead in our implement.
3) **Trade-off between the efficiency and the precision:** Just a little modification, an approximate nearest neighbor algorithm based on M2M can be designed.
4) **Trade-off between time efficiency and space cost:** There is several ways to reduce the space cost, such as reduce the number of level.
5) **Extend algorithm for high dimension:** it wouldn't be difficult to Extend M2M algorithm for high dimension, which can refer the theorems in the paper of Bentley et al [5].

M2M-NN or even M2M model are newborn. There is still much room for improvement. And these require a long-term task for the researchers.

REFERENCES

[1] SHAMOS, M.I, AND HOEY, D. Closest-point problems Proc 16th IEEE Syrup. Foundatmns of Computer Scwnce, pp. 151-162, Oct 1975.

[2] RABIN, M O. Probabilistic algorithms, in Algorithms and Complexity: New Dwectmns and Recent Results, J.F. Traub (Ed.), Academic Press, New York, pp. 21-39, 1976.

[3] WEIDE, B.W. Statistical methods m algorithm design and analysis. Ph.D. Dissertation, Carnegm-Mellon Umv, Pittsburgh, Pa (Appeared as CMU Comput Sci. Rep. CMU-CS-78-142), Aug 1978

[4] FORTUNE, S., AND HOPCROFT, J E A note on Rabln's nearest-neighbor algorithm. Inf. Process. Lett. 8, 1 , 20-23, Jan. 1979.

[5] BENTLEY, J. L.,WEIDE, B. W., AND YAO, A. C. Optimal expected-time algorithms for closest point problems. ACM Trans. Math. Softw. 6, 4, 563–580,1980

[6] BERN, M. 1993. Approximate closest-point queries in high dimensions. Inf. Proc. Lett. 45, 95–99.

[7] BERN, M., EPPSTEIN, D., AND TENG, S.-H. 1993. Parallel construction of quadtrees and quality triangulations. In Proceedings of the 3rd Workshop Algorithms Data Structures. Lecture Notes in Computer Science, vol. 709. Springer-Verlag, New York, pp. 188 –199.

[8] J. H. Friedman, J. L. Bentley, and R. A. Finkel. An algorithm for finding best matches in logarithmic expected time. ACMTransactions on Mathematical Software, 3(3):209–226, September 1977.

[9] S. Arya, D. Mount, N. Netanyahu, R. Silverman, and A. Wu. An Optimal Algorithm for Approximate Nearest Neighbor Searching Fixed Dimensions. Journal of the ACM, 45(6):891–923,1998.

[10] An intoductory tutorial on kd-trees, Extract from Andrew Moore's PhD Thesis: Ecient Memory-basedd Learning for Robot Control PhD. 1991

[11] T. Liu, A. W. Moore, and A. Gray. Efficient exact k-NN and nonparametric classification in high dimensions. In S. Thrun, L. Saul, and B. Sch¨olkopf, editors, Advances in Neural Information Processing Systems 16. MIT Press, Cambridge, MA, 2004

[12] T.H. Cormen, C.E. Leiserson, R.L. Rivest, and C. Stein, Introduction to Algorithms (2nd E.), MIT Press, McGraw-Hill, New York, USA, 2001.

[13] Sebastian Nowozin. A vanilla k-d tree implementation 2004.