A New K Nearest Neighbours Algorithm Using Cell Grids for 3D Scattered Point Cloud

Jianhui Zhao[1], Chengjiang Long[1], Shuping Xiong[2], Cheng Liu[1], Zhiyong Yuan[1]
[1]*School of Computer, Wuhan University,*
Wuhan, Hubei, 430072, P. R. China
[2]*School of Design and Human Engineering, Ulsan National Institute of Science and Technology,*
Ulsan, 689–798, Republic of Korea
jianhuizhao@whu.edu.cn

***Abstract*—In this paper, we propose one novel and efficient K nearest neighbours search algorithm based on 3D uniform cell grids. First, a simple min-max box is used to store all the points and a twice division strategy is adopted to determine the edge length of basic grids. Then we limit the search space for each grid with certain query points to control the amount of distance calculations under a suitable level. And the computational cost of sorting operations is reduced effectively through avoiding the unnecessary calculations, with the help of properly determined subspaces and sphere spaces for points. Compared with existing related algorithms, our method can search for the K nearest neighbours accurately and quickly, and it has many possible applications in the fields using 3D scattered point clouds.**

***Index Terms*—K nearest neighbours, search algorithm, 3D point cloud, cell grids.**

I. INTRODUCTION

Point primitives have been used popularly as one kind of surface representation in the fields of virtual reality, scientific visualization, geometric modelling, reverse engineering, and so on [1]–[4]. With the advanced scanning technologies, it is possible for lots of real objects to be scanned into 3D point clouds [5]. Note that the point cloud is nothing more than 3D coordinates of a collection of scanned points, and contains no topological information [6]. But other additional geometric information on the surface, such as surface normal or local curvature [7] which have to be calculated from the positional data, are required necessarily for the subsequent processing [8]. What is common to obtain local geometric information from the point cloud is to calculate K nearest neighbours for each point [9]. Getting the neighbours is meaningful. First, the calculation can be as small as possible to allow efficient processing of mass data sets. Second, neighbours guarantee that their local geometric information can be approximated provably well to a certain extent.

Aiming at increasing search speed, many research works have been performed consequently, and they are developed based on the basic idea: all the scattered points are firstly divided into small "regions" and stored into the related spatial data structure, then some strategies are used to find K nearest neighbours for the query point. There are different kinds of shapes already been adopted for the small regions, including Voronoi diagram [10], [11], hyper rectangular bucket [12]–[14] (e.g. K-d tree, Quadtree and Octree), bounding rectangle [15], [16] (e.g. R-tree and SR-tree), bounding sphere [17] (e.g. SS-tree), pyramid [18] and uniform cell grids [19], [20].

All the methods intended to set up reasonable and efficient data storage structures to search for the K nearest neighbours of query point with higher speed. However, the calculation of Voronoi diagram is still unbearable especially for the large scattered point sets. Comparatively, other data structures are more effective and acceptable. Bentley's method of K-d tree [21] has been widely used. It partitions the data space into hyper-rectangular buckets, then K nearest neighbours of any query point are obtained with a binary search for the target bucket and a local search for desired points in target bucket and its neighbouring buckets. A fast K nearest neighbours algorithm proposed by Sankaranarayanan [22] identifies a region in space that contains all of the K nearest neighbours for a collection of points. Each query point searches only the locality for the correct set of K nearest neighbours once the best possible locality is built. What's more, any hierarchical spatial data structure can be used in this method including some structures that are based on object hierarchies such as R-tree, Quadtree and Octree.

Different with the above methods, the algorithm presented in our paper divides the point cloud data into the basic 3D uniform cell grids using a twice division strategy. Based on the adjacent relationship between arbitrary 3D cell grid and the grids around it, related subspaces and sphere spaces are defined with the outwards expansion of cell grid, and they are used to limit the search space for each grid. Through reducing the expense of sorting operations, our approach can obtain the K nearest neighbours with less time.

The rest of our paper is organized as: the preliminary work to establish the 3D cell grids are presented in Section II, the proposed novel search algorithm is described in Section III, experimental results are given and analysed in Section IV, and finally the conclusion is drawn in Section V.

Manuscript received March 28, 2013; accepted October 24, 2013.
This research was funded by National Basic Research Program of China (973 Program, No. 2011CB707904), National Natural Science Foundation of China (No. 60603079, 61272276), Science and Technology Bureau of Suzhou of China (No. SH201115).

II. PRELIMINARY

Before searching for K nearest neighbours in a set of point cloud, the 3D cell grids need to be established to divide all scattered points into related units including two types: empty grids and non-empty grids, while each non-empty cell grid contains at least one point.

The minimum and maximum values of x, y, z coordinates of all points are denoted as $x_{\min}$, $y_{\min}$, $z_{\min}$ and $x_{\max}$, $y_{\max}$, $z_{\max}$ respectively. Then length, width and height of the cuboid containing the point cloud are calculated as:

$$a = x_{\max} - x_{\min}, \tag{1}$$

$$b = y_{\max} - y_{\min}, \tag{2}$$

$$c = z_{\max} - z_{\min}. \tag{3}$$

For the cuboid with volume $V = abc$, in ideal situation the scattered points are distributed in its 3D space uniformly. Suppose the space is well divided so that each grid contains one point, the edge length of each grid can be preliminarily set as $L_0 = \sqrt[3]{\frac{V}{N}}$. In this case, there are $l_0 \times m_0 \times n_0$ divided cell grids, where:

$$l_0 = \left\lceil \frac{a}{L_0} \right\rceil, \tag{4}$$

$$m_0 = \left\lceil \frac{b}{L_0} \right\rceil, \tag{5}$$

$$n_0 = \left\lceil \frac{c}{L_0} \right\rceil. \tag{6}$$

In fact, the scattered point cloud is seldom well distributed as above. Therefore, suppose N_{cube} is the total number of 3D cell grids that actually contain at least one point, the average density of points in cuboid is computed as $\circ = \frac{N}{N_{cube} \times L_0^3}$.

Since our algorithm is designed to search for K neighbours of every point, we can make a necessary adjustment based on the average density for the edge length of each grid as

$$L = \sim \times \sqrt[3]{\frac{K}{\circ}}, \tag{7}$$

where $\sim$ is the adjusting parameter, and its value may vary for different kinds of scattered point clouds.

Then the points are divided for the second time and are re-distributed into $l \times m \times n$ grids, where:

$$l = \left\lceil \frac{a}{L} \right\rceil, \tag{8}$$

$$m = \left\lceil \frac{b}{L} \right\rceil, \tag{9}$$

$$n = \left\lceil \frac{c}{L} \right\rceil. \tag{10}$$

Now, any cell grid in the cuboid can be denoted with the index number (i, j, k), which is equivalent to the following descriptions:

$$x = x_{\min} + iL, \tag{11}$$

$$y = y_{\min} + jL, \tag{12}$$

$$z = z_{\min} + kL, \tag{13}$$

where (x, y, z) is the grid vertex with minimum coordinates.

III. OUR SEARCH ALGORITHM

A. Related Definitions

Definition (1), Subspace:

Let the space $\{(i, j, k) \mid \max(|i - i_0|, |j - j_0|, |k - k_0|) \le r\}$ be a subspace of the grid (i_0, j_0, k_0) and denoted as $S(r)$, which is obtained by the extension outwards with r times of the edge length L from the cell grid (i_0, j_0, k_0). If the minimum and maximum grid vertices of the subspace belong to grid (i_1, j_1, k_1) and grid (i_2, j_2, k_2) respectively, subspace $S(r)$ can be also denoted as $(i_1, j_1, k_1, i_2, j_2, k_2)$.

Definition (2), Ultimate space:

After extension outwards for s times with edge length L from the grid (i_0, j_0, k_0), if subspace $S(s)$ contains the K nearest neighbours of an arbitrary point in grid (i_0, j_0, k_0) for the first time, subspace $S(s)$ is an ultimate space and denoted as $S_{ultimate}$. The ultimate space is used to limit the search range for any query point in grid (i_0, j_0, k_0).

Definition (3), Sphere space:

Given one query point O in grid (i_0, j_0, k_0), the sphere with radius R centred with O is the sphere space of point O, and is denoted as S_{sphere}.

For any query point in a non-empty cell grid, two kinds of its S_{sphere} are introduced: internal sphere space $S_{insphere}$, and external sphere space $S_{outsphere}$. The radius of $S_{insphere}$ and the radius of $S_{outsphere}$ are denoted as R_{in} and R_{out} respectively, where R_{in} is less than R_{out}.

To illustrate the relationships among $S_{ultimate}$, $S_{outsphere}$ and $S_{insphere}$, a 2D sectional view is shown in Fig. 1. The query point O is marked with "+", and its corresponding cell grid is marked with the smallest square. $S_{ultimate}$ of the grid is represented by the outermost square, $S_{outsphere}$ of the point is represented by the outer circle, while $S_{insphere}$ of the point is shown with the inside circle. Also, we denote the shadow area as $S_{outsphere} - S_{insphere}$, the difference between these two sphere spaces for the same query point.

The idea to adopt $S_{insphere}$ and $S_{outsphere}$ for query point O is: checking all the points in $S_{ultimate}$, if the number of points in $S_{insphere}$ is no more than K, all these points are taken as neighbours of query point O, and the rest of

neighbours are selected through sorting the points in space $S_{outsphere} - S_{insphere}$; otherwise, the K nearest neighbours are obtained by sorting points within the space of $S_{insphere}$.

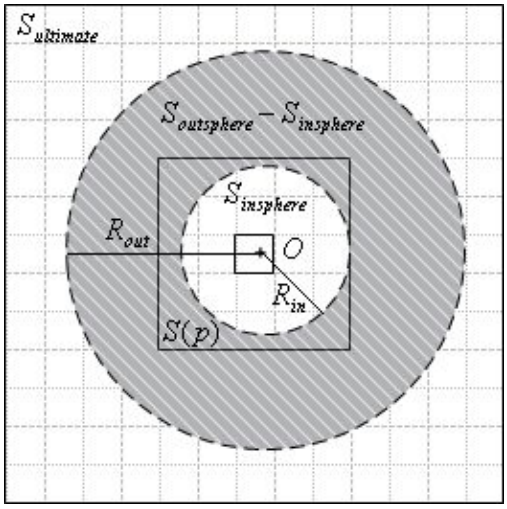


Fig. 1. Relationships of $S_{ultimate}$, $S_{outsphere}$ and $S_{insphere}$.

Since the cost of sorting algorithm in 1D space is ¦ log ¦ (¦ is the scale of space), searching algorithm of K nearest neighbours for 3D space requires much more number of sorting operations. Therefore, choosing suitable values of $S_{insphere}$, $S_{outsphere}$ and $S_{ultimate}$ is very helpful to control the number of basic operations, and is thus crucial for our algorithm.

B. Determinations of $S_{insphere}$, $S_{outsphere}$ and $S_{ultimate}$

Assume query point O is located at an arbitrary position in grid (i_0, j_0, k_0), we can extend the grid outwards gradually until subspace $S(p)$ (shown in Fig. 1), which is extended for p times of edge length L and contains no less than K points for the first time. Also we denote dx, dy, dz as the larger distances from query point O to the two surfaces of its grid (i_0, j_0, k_0) along x-axis, two surfaces along y-axis, and two surfaces along z-axis respectively. If the coordinates of query point O are (x_o, y_o, z_o), there are:

$$dx = \max(x_o - i_0 L, (i_0 + 1)L - x_o), \quad (14)$$

$$dy = \max(y_o - i_0 L, (i_0 + 1)L - y_o), \quad (15)$$

$$dz = \max(z_o - i_0 L, (i_0 + 1)L - z_o). \quad (16)$$

(1) Determining of $S_{insphere}$

Suppose d_{short} is the minimum value of shorter distances between query point O and the two surfaces of its grid along x-axis, y-axis and z-axis, it can be described as

$$d_{short} = \min(L - dx, L - dy, L - dz). \quad (17)$$

Thus the radius R_{in} of $S_{insphere}$ can be calculated as

$$R_{in} = pL + d_{short}. \quad (18)$$

Obviously, if the number of points in the internal sphere space $S_{insphere}$ of the query point O is less than or equal to K, they must all belong to the K neighbours of point O without taking any sorting operations.

(2) Determining of $S_{outsphere}$

Since subspace $S(p)$ has expanded p times of the edge length L from grid (i_0, j_0, k_0), the radius R_{out} of $S_{outsphere}$ can be described as

$$R_{out} = \sqrt{(dx + p \times L)^2 + (dy + p \times L)^2 + (dz + p \times L)^2}. \quad (19)$$

From (14)–(16), it is easy to know that values of dx, dy, dz are less than edge length L. Thus the distance d from the query point O to any other point in $S(p)$ must be satisfied with

$$d \le R_{out} \le (p+1)\sqrt{3}L. \quad (20)$$

Since there are at least K points in subspace $S(p)$, the number of points in $S_{outsphere}$ must be more than or equal to K, i.e. all the K nearest neighbours of query point O are located in space $S_{outsphere}$.

(3) Determining of $S_{ultimate}$

Similar with subspace $S(p)$, subspace $S(\lfloor (p+1)\sqrt{3} \rfloor + 1)$ is obtained after expanding the 3D cell grid (i_0, j_0, k_0) for $(\lfloor (p+1)\sqrt{3} \rfloor + 1)$ times of edge length L, and

$$S(p) \subset S(\lfloor (p+1)\sqrt{3} \rfloor + 1). \quad (21)$$

Since $S(p)$ contains at least K nearest neighbour points for query point O, for any arbitrary point in the same grid (i_0, j_0, k_0), its K nearest neighbours can be limited in space $S(\lfloor (p+1)\sqrt{3} \rfloor + 1)$, which is the $S_{ultimate}$ of all points in the 3D cell grid (i_0, j_0, k_0).

The mechanics of our algorithm are shown in Fig. 1. For example, if $p = 2$, $S_{ultimate}$ of the related grid (i_0, j_0, k_0) is set as $S(\lfloor (2+1)\sqrt{3} \rfloor + 1) = S(6)$. For query point O in cell grid (i_0, j_0, k_0), all the K nearest neighbours can be obtained from its corresponding spaces $S_{insphere}$ and $S_{outsphere}$.

C. Implementation of Searching Algorithm

For 3D scattered points, there are usually more than one point in most of non-empty cell grids, as shown in Fig. 2. Of course, all the points in the same cell grid share the common $S(p)$ and $S_{ultimate}$.

When searching for K nearest neighbours for the first point in cell grid, we record $S_{ultimate}$ and p. Then the other points in the same grid can directly use the value of p to compute both R_{in} of $S_{insphere}$ and R_{out} of $S_{outsphere}$ without

extending subspace from the original grid gradually and repeatedly. In this way, we can avoid a large number of unnecessary calculations, and thus improve the search efficiency.

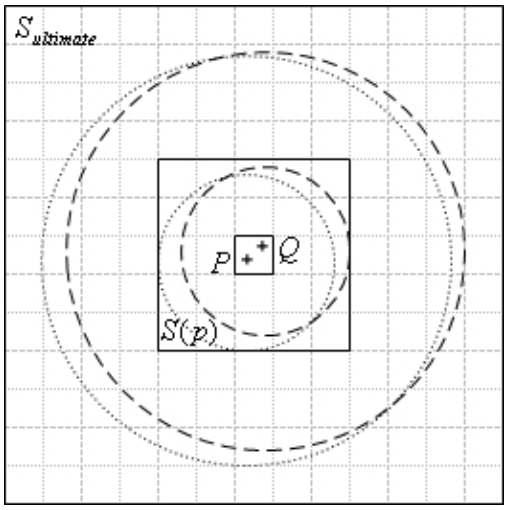


Fig. 2. Points P and Q in the same 3D cell grid.

The specific steps of our algorithm are as follows:

Step 1. Calculate the edge length L of 3D cell grid and divide all points into the grids;

Step 2. Traverse all the scattered points in point cloud, for any query point P, if the cell grid with point P has already been visited, go to Step 3; otherwise calculate the subspace $S_{ultimate}$ and record the intermediate value p, then set status of the cell grid as visited;

Step 3. Calculate radius R_{in} of $S_{insphere}$ and radius R_{out} of $S_{outsphere}$ for the query point P;

Step 4. Count the number of points (denoted as a) in subspace $S_{insphere}$ for query point P:

– If $a < K$, all the points in subspace $S_{insphere}$ belong to K neighbours of point P, and the remaining $(K - a)$ neighbours are selected through sorting operations from the points in space $S_{outsphere} - S_{insphere}$;

– If $a = K$, all the points in subspace $S_{insphere}$ are just the K neighbours of point P;

– If $a > K$, all the K neighbours of point P must be located in subspace $S_{insphere}$, and they can be selected through sorting operations from the points in $S_{insphere}$.

Step 5. Go to Step 2, until all the scattered points in point cloud have been visited and processed.

IV. Experimental Results

Experiments are conducted to evaluate the performance of our algorithm, and the experimental environment is Intel(R) Core(TM) 2 Duo CPU E7400 processor 2.80 GHz, 2.0 GB RAM, Windows XP Operation System, and Microsoft Visual Stdio.Net 2008 IDE. The employed 3D scattered point clouds for experiments are often used in computer graphics or other relevant fields, and the number of points in these 3D scattered point clouds range from 10 k to 120 k.

According to [22], two parameters are usually applied to compare the effects of different kinds of K nearest neighbours searching algorithms: the executed time, and the Euclidean distance sensitivity which is defined and calculated with the following

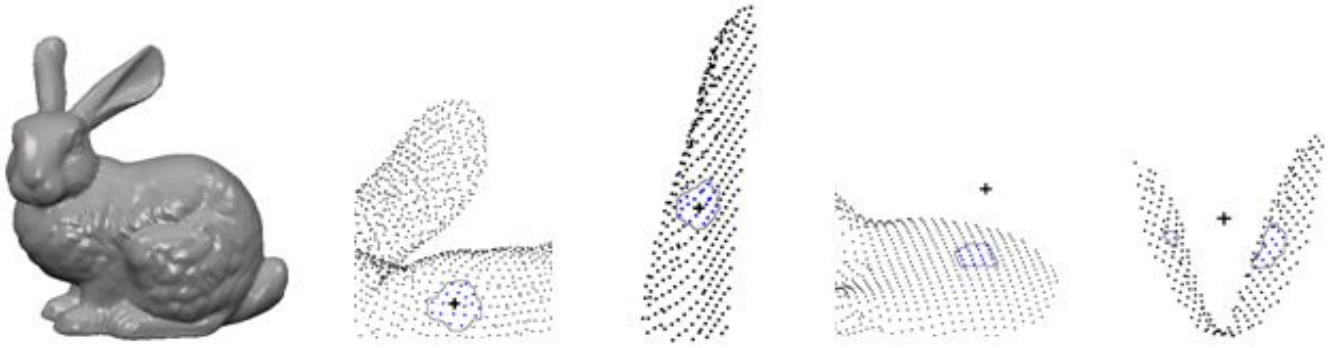
$$\overset{\Omega}{S} = \frac{N(dist\_calc)}{N \log N}, \tag{22}$$

where $\overset{\Omega}{S}$ and $N(dist\_calc)$ represent distance sensitivity and total number of point distance calculations (the most time consuming operations for sorting points) respectively, while $N \log N$ is the cost of sorting N points.

A. Verification of Searching Accuracy

To verify the accuracy of our new searching algorithm for K nearest neighbours, the Stanford Bunny model containing 35,947 points is taken as test data set, as shown in Fig. 3. We consider all possible cases of query points, which are divided into four categories: (a) the query point is one of the smooth points sampled from the back of bunny; (b) the query point is one of the ridge points sampled from the ear of bunny and the curvature of these points changes significantly; (c) the query point is one man-made point, which is far from the back of bunny; (d) the query point is one man-made point, which is located between two ears of bunny.

Figure 3 illustrates the above four categories of query points selected from the bunny model and marked by "+". K is set as 20, and the corresponding 20 nearest neighbours are searched for every query point. To show them clearly, the searched K neighbour points are circled by curves.



a) b) c) d) e)

Fig. 3. Different types of query points and their K nearest neighbours.

B. Adjustment of Parameter ~

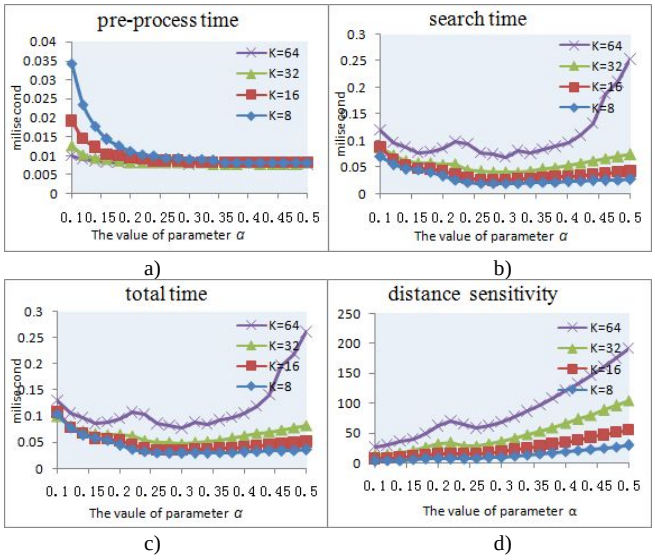


Fig. 4. The effects from parameter ~ on (a) pre-processed time, (b) searched time, (c) total time, and (d) Euclidean distance sensitivity for different values of K.

Parameter ~ in (7) is the key parameter of our algorithm, since it affects the edge length of 3D cell grids and thus influences the following step of subspace expansion. So we study the effects of different ~ on the performance of the proposed K nearest neighbours' algorithm with the 3D point cloud of Stanford Bunny model. For a given integer value of K, i.e. 8, 16, 32, 64, the corresponding value of ~ varies from 0.10 to 0.50. Performance of our algorithm is evaluated through measuring the executed time (including pre-process time, searched time, and the sum of them, i.e. total time) and the value of Euclidean distance sensitivity. The pre-process time is the period used to read point cloud from model file and then divide the points into corresponding spatial data storage structure, i.e. the 3D cell grids; the searched time is the period used to search for the K nearest neighbours based on the cell grids; while the total time is the whole period including the pre-process time and the searched time.

Figure 4(a) shows the effects of ~ on the pre-process time for different K values. When ~ is a larger value, the pre-process time is less. Since larger ~ produces bigger 3D each grid, and thus the division time for points is less. Figure 4(b) shows the effects of ~ on searched time. If ~ is too small, there will be a huge number of small 3D cell grids to be processed; if ~ is too big, the cost for each big grid will be increased obviously. When ~ is between 0.25 and 0.3, the curves display higher searching efficiencies. Figure 4(c) shows the effects of ~ on total time, where the curve shapes are nearly the same as those of Fig. 4(b), due to the fact that pre-process time is only a small proportion in the total time of our algorithm. Figure 4(d) shows the effects of ~ on distance sensitivity, and it can be found that the distance sensitivity is relatively low when ~ is very small or varies from 0.22 to 0.26.

Then we give further analysis on the relationships between parameter ~ and the average numbers of points in $S_{ultimate}$, $S_{insphere}$, $S_{outsphere}$, and average number of nearest neighbours obtained directly from the points in $S_{insphere}$ without taking any sorting operations, as illustrated in Fig. 5.

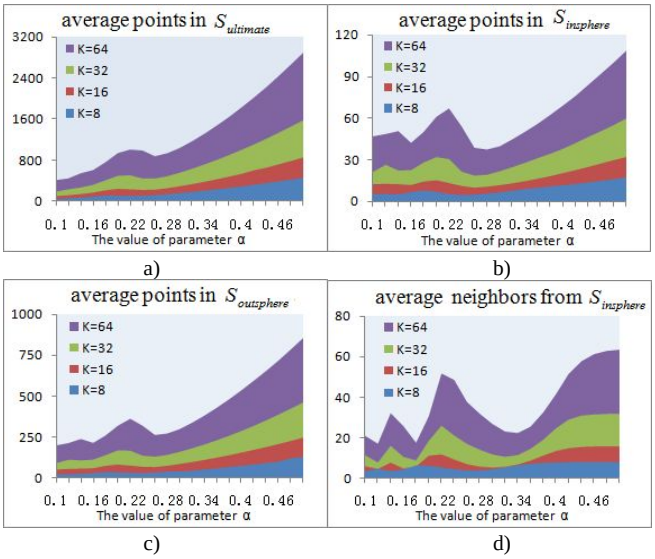


Fig. 5. The effects from parameter ~ on (a) average points in $S_{ultimate}$, (b) average points in $S_{insphere}$, (c) average points in $S_{outsphere}$, and (d) average number of nearest neighbours found directly from $S_{insphere}$ for different values of K.

Figure 5(a) shows that when ~ varies round 0.26, the average points in $S_{ultimate}$ can be controlled under a relatively stable and low number; Fig. 5(b) and Fig. 5(c) show the similar varying tendency as Fig. 5(a), since ~ affects the spaces $S_{ultimate}$, $S_{insphere}$ and $S_{outsphere}$ in a similar way; Fig. 5(d) shows that when ~ varies between 0.20 and 0.26, a higher number of average nearest neighbours can be directly obtained from the space $S_{insphere}$ without any sorting operations, which is very helpful to avoid the unneeded computations and thus improve the searching efficiency of our algorithm.

From the experiments it can be found that parameter ~ indirectly affects the speed to search for K nearest neighbours through the edge length of 3D cell grids. If ~ is very large, although $S_{ultimate}$ can be determined quickly, both $S_{insphere}$ and $S_{outsphere}$ contain excessive numbers of points, and thus will decrease the search efficiency. On the contrary, if ~ is very small, although $S_{insphere}$ and $S_{outsphere}$ contain less numbers of points, the number of cell grids will be huge due to the small edge length of basic cell grids, accordingly expansion speed of the subspaces will also be slow.

Therefore, to guarantee the higher search efficiency, there should be a suitable value of ~ . Based on aforementioned analysis of the experimental results, in our work parameter ~ is set between 0.22 and 0.30.

C. Comparison with Related Algorithms

Our algorithm is further evaluated through comparing both executed time and distance sensitivity with the other existing related methods, i.e. Bentley's method [21] (represented with Bent's) and Sankaranarayanan's method [22] (represented by Sank's), which are popularly used.

a) b) c) d) e)

Fig. 6. Point clouds of five 3D models: foot, bust, golf, horse and teeth (from left to right).

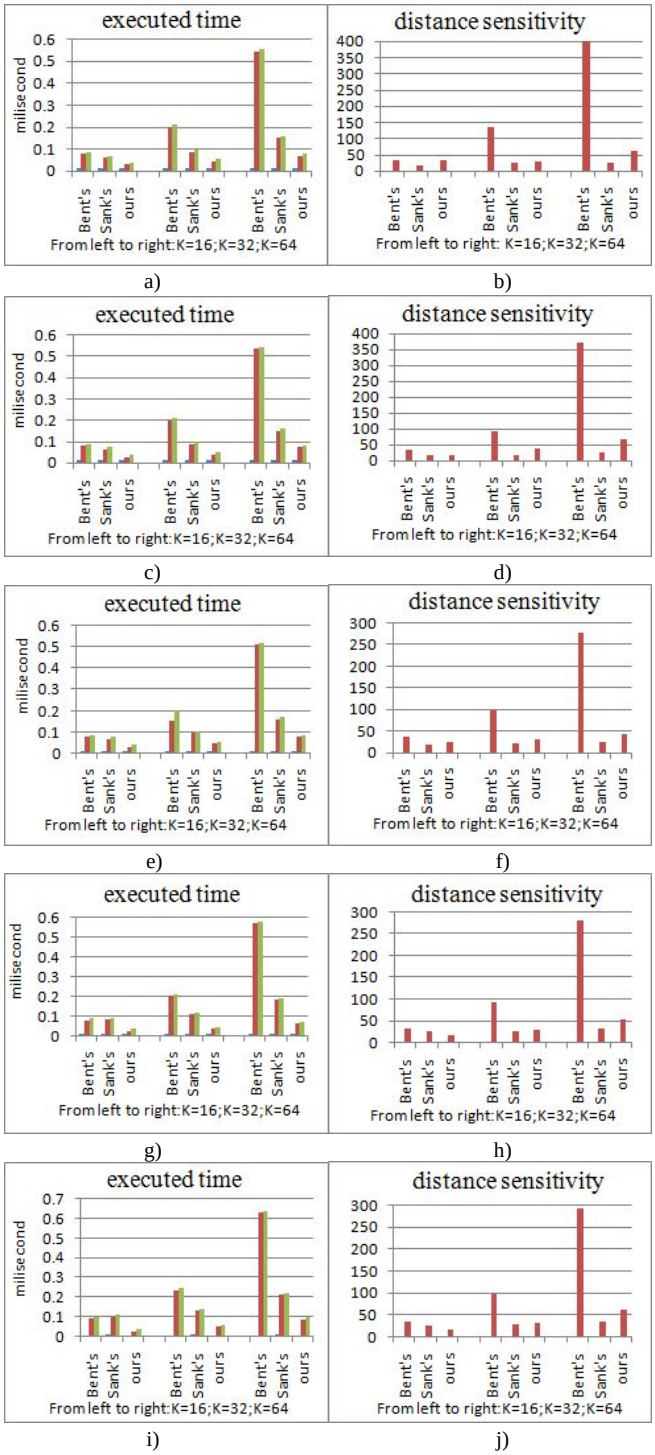


Fig. 7. Comparisons for the executed time and the distance sensitivity of three algorithms: Bent's, Sank's and Ours, on the point clouds of five 3D models (foot (a), (b); bust (c), (d); golf (e), (f); horse (g), (h); teeth (i), (j)).

The point clouds of five 3D models are employed for experiments, and they are: foot with 10016 scanned points, bust with 25126 scanned points, golf with 39479 scanned points, horse with 48485 scanned points, teeth with 117871 scanned points, as shown in Fig. 6. During experiments, different values of K, i.e. 16, 32, 64, are adopted to obtain the comprehensive comparisons among the three searching methods.

The experimental results are shown in Fig. 7, where each row displays the results from one of the five 3D models. The executed time includes pre-processed time, searched time and total time. They are illustrated by blue, red and green bars respectively, while the height of green bar is the sum of the heights of blue bar and red bar.

It can be found from Fig. 7 that the pre-processed times for three methods have only slight differences, but our method has the least searched time and total time, i.e. having the best search efficiency. As for Euclidean distance sensitivity, our method is superior to Bent's method and inferior to Sank's method. The reason is that we have to perform many distance calculations to obtain $S_{ultimate}$, $S_{insphere}$, $S_{outsphere}$. But these computations are necessary since they are very helpful to reduce the searched time in the following steps, and achieve the final results with less total time.

V. CONCLUSIONS

In this paper, a novel search algorithm is proposed for K nearest neighbours based on 3D cell grids. The contributions of our approach include: (1) we use a simple min-max box composed of 3D uniform cell grids to store all the scattered points, and adopt the twice division strategy to determine the edge length of the cell grids in order to control the step of extension outwards from the original query cell grid; (2) to control the distance calculations under a suitable level, we introduce the "ultimate space" to limit the search space for each cell grid with certain query points; (3) to reduce the scale of sorting operations for scattered points, we define the sphere spaces including "internal sphere space" and "external sphere space", from them the K nearest neighbours can be obtained while unneeded sorting operations are avoided; (4) for the situation that there are more than one point in the same cell grid, we only need to calculate the "ultimate space" of the cell grid for one time, and the corresponding values can be recorded to avoid repeated calculations when all the points in the same grid are considered as query points.

Experimental results have proved the accuracy of our new algorithm. Based on statistical analysis, setting of parameter is discussed. Through comparisons with the existing related methods, our algorithm has shown the advantages of owning

much better search efficiency while controlling the distance sensitivity under a suitable level.

REFERENCES

[1] V. Matiukas, D. Miniotas, "Point cloud merging for complete 3D surface reconstruction", *Elektronika ir Elektrotechnika (Electronics and Electrical Engineering)*, no. 7, pp. 73–76, 2011.

[2] L. Kobbelt, M. Rotsch, "A survey of point-based techniques in computer graphics", *Computers & Graphics*, no. 28, pp. 801–804, 2004. [Online]. Available: http://dx.doi.org/10.1016/j.cag.2004.08.009

[3] Y. L. Chen, S. H. Lai, "An orientation inference framework for surface reconstruction from unorganized point clouds", *IEEE Trans. Image Processing*, vol. 20, no. 3, pp. 762–775, 2011. [Online]. Available: http://dx.doi.org/10.1109/TIP.2010.2076297

[4] J. Yang, L. Zhao, "Research on 3D reconstruction of the facial surface reverse engineering", *Advanced Manufacturing Systems*, no. 201–203, pp. 113–116, 2011.

[5] G. Rainer, E. Bjoern, S. Heiko, "Comparison and classification of 3D objects surface point clouds on the example of feet", *Machine Vision and Applications*, vol. 22, no. 2, pp. 235–243, 2011. [Online]. Available: http://dx.doi.org/10.1007/s00138-009-0230-y

[6] D. Zhang, L. Wang, J. Yu, "Geometric topology based cooperation for multiple robots in adversarial environments", *Control Engineering Practice*, vol. 16, no. 9, pp. 1092–1100, 2008. [Online]. Available: http://dx.doi.org/10.1016/j.conengprac.2007.12.004

[7] J. Wu, W.A.P. Smith, E.R. Hancock, "Gender discriminating models from facial surface normals", *Pattern Recognition*, vol. 44, no. 12, pp. 2871–2886, 2011. [Online]. Available: http://dx.doi.org/10.1016/j.patcog.2011.04.013

[8] T. R. Jones, F. Durand, M. Desbrum, "Non-iterative, feature-preserving mesh smoothing", *ACM Trans. Graphics*, vol. 22, no. 3, pp. 943–949, 2003. [Online]. Available: http://dx.doi.org/10.1145/882262.882367

[9] N. Amenta, M. Bern, M. Kamvysselis, "A New Voronoi-Based Surface Reconstruction Algorithm", in *1998 Proc. of the 25th Annual ACM Conference on Computer Graphics and Interactive Techniques*, pp. 415–421.

[10] M. T. DieKerson, R. L. S. Drysdale, J. R. SacK, "Simple algorithms for enumerating inter-point distances and finding K nearest neighbours", *International Journal of Computational Geometry and Applications*, vol. 2, no. 3, pp. 221–239, 1992. [Online]. Available: http://dx.doi.org/10.1142/S0218195992000147

[11] D. Kim, D. S. Kim, "Region-expansion for the Voronoi diagram of 3D spheres", *Computer-Aided Design*, vol. 38, no. 5, pp. 417–430, 2006. [Online]. Available: http://dx.doi.org/10.1016/j.cad.2005.11.007

[12] J. Yuan, "A does distribution evaluation technique using the k-d tree for nearest neighbour searching", *Medical Physics*, vol. 37, no. 9, pp. 4868–4873, 2010. [Online]. Available: http://dx.doi.org/10.1118/1.3480964

[13] A. O. Finley, R. E. McRoberts, "Efficient k-nearest neighbour searches for multi-source forest attribute mapping", *Remote Sensing of Environment*, no. 112, pp. 2203–2211, 2008. [Online]. Available: http://dx.doi.org/10.1016/j.rse.2007.08.024

[14] H. Samet, *Foundations of Multidimensional and Metric Data Structures*. Morgan-Kaufmann, San Francisco, CA, 2006.

[15] Y. Gao, B. Zheng, G. Chen, W. C. Lee, K. C. K. Lee, Q. Li, "Visible reverse nearest neighbour queries", in *Proc. IEEE 25th Int. Conf. Data Engineering*, 2009, pp. 1203–1206.

[16] R. H. Guting, T. Behr, J. Xu, "Efficient k-nearest neighbour search on moving object trajectories", *The VLDB Journal*, vol. 19, no. 5, pp. 687–714, 2010. [Online]. Available: http://dx.doi.org/10.1007/s00778-010-0185-7

[17] N. Katayama, S. Satoh, "The SR-tree: an index structure for high-dimensional nearest neighbour queries", in *Proc. ACM SIGMOD Int. Conf. Management of Data*, 1997, pp. 369–380.

[18] S. Berchtold, C. Bohm, H. P. Kriegel, "The pyramid-technique: towards breaking the curse of dimensionality", in *Proc. ACM SIGMOD Int. Conf. Management of Data*, 1998, pp. 142–153.

[19] L. A. Piegl, W. Tiller, "Algorithm for finding all k nearest neighbours", *Computer-Aid Design*, vol. 34, pp. 167–172, 2002. [Online]. Available: http://dx.doi.org/10.1016/S0010-4485(00)00141-X

[20] R. Liang, C. Chen, Z. Pan, H. Zhang, "A fast surface reconstruction algorithm based on 3D point set", *Journal of Image and Graphics*, vol. 8A, no. 1, pp. 63–67, 2003.

[21] J. L. Bentley, "K-d trees for semidynamic point sets", in *Proc. of the 6th Annual ACM Symposium on Computational Geometry*, 1990, pp. 187–197.

[22] J. Sankaranarayanan, H. Samet, A. Varshney, "A fast all nearest neighbour algorithm for applications involving large point-clouds", *Computers & Graphics*, vol. 31, pp. 157–174, 2007. [Online]. Available: http://dx.doi.org/10.1016/j.cag.2006.11.011