VECIMS 2009 - International Conference on Virtual Environments, Human-Computer Interfaces and Measurements Systems
Hong Kong, China
May 11-13, 2009

Quadtree Based Mouse Trajectory Analysis for Efficacy Evaluation of Voice-enabled CAD

X.Y. Kou, X.C. Liu, S.T. Tan
Department of Mechanical Engineering
The University of Hong Kong
Hong Kong, China
kouxy@hku.hk, no1rr@hku.hk,

***Abstract*— Voice-enabled applications have caught considerable research interest in recent years. It is generally believed that voice based interactions can improve the working efficiencies and the overall productivities. Quantitative evaluations on the performance boost by using such Human-Computer interactions (HCI) are therefore necessary to justify the claimed efficacies and the usefulness of the HCI system. In this paper, a quadtree based approach is proposed to analyze the mouse movement distributions in the proposed Voice-enabled Computer-Aided Design (VeCAD) system. The mouse tracker keeps a record of all the mouse movement during the solid modeling process, and a quadtree based approach is applied to analyze the mouse trajectory distributions in both the traditional CAD and the VeCAD system. Our experiments show that the mouse movement is significantly reduced when voice is used to activate CAD modeling commands.**

Quadtree; Voice; CAD; Mouse trajectory; HCI; Solid modelling

I. INTRODUCTION

With the recent development in speech synthesis and recognition technologies [1-3], voice based human-computer-interaction (HCI) has been extensively used in a variety of applications such as word processing [4], medical, biomedical [5] and telephony [6] areas. Using voice input and accurate speech recognition, routine operations can be efficiently executed with intuitive voice commands, and the ease of use of a system can be improved.

The integration of human voice into a Computer-Aided Design (CAD) system has been investigated for decades [7-12]. Among these efforts, simple voice commands were first used to construct and manipulate primitive solid models, for instance block constructions or object rotations. The vocabulary used was rather restrictive initially, and the roles that voice interaction played were limited. In these systems, voice was mostly used in combination with other means of interactions, for instance, data gloves [7, 8, 11, 12] and eye trackers [12] were used to cooperate with voice to perform solid modeling operations. Voice-enabled CAD (VeCAD) systems capable of designing complex solid models have been reported recently, where voice is no longer considered as an auxiliary mean of interactions, but rather can undertake a majority of tasks [13-16] such as command firing and object dimensioning.

It is generally believed that voice based interaction can improve design efficiencies and promote the overall productivities of a CAD system. However, it is often implicit or unclear, to what degrees can the efficiencies be boosted in comparison with the direct (i.e. mouse and keyboard based) interactions.

This paper is motivated to quantitatively evaluate on the performance enhancement using a voice-enabled CAD system. The idea is to use the mouse movement as a quantitative measure, and compare their location distributions using and without using voice in the CAD modeling process. The same model construction tasks are separately conducted using the VeCAD and traditional CAD system. A mouse tracker is designed to keep a record of all the mouse movement during a solid modeling process. A quadtree based mouse trajectory analysis approach is then proposed to efficiently query the mouse trajectory distributions.

II. QUADTREE BASED MOUSE TRAJECTORY ANALYSIS

A. Problem statement

The problem under investigation is to get the quantitative statistics of the mouse movement in typical solid modeling processes, where voice based interactions are either enabled or disabled. In our voice-enabled CAD system [13, 14], voice is used to activate 500+ commonly used modeling commands such as sketching, dimensioning and feature generation. The VeCAD is designed to free designers from tedious command searching via menu clicks, which is increasingly arduous if a complex CAD system with a large number of commands is used.

Specifically, the following queries of mouse movement are of particular interest:

1. How many times is the mouse moved to a particular location (x, y)?

978-1-4244-3809-9/09/$25.00 ©2009 IEEE

2. To accomplish a CAD modeling task, how many mouse moves are used solely to activate a CAD command?
3. To what degree can voice reduce the mouse movement and improve the performance?

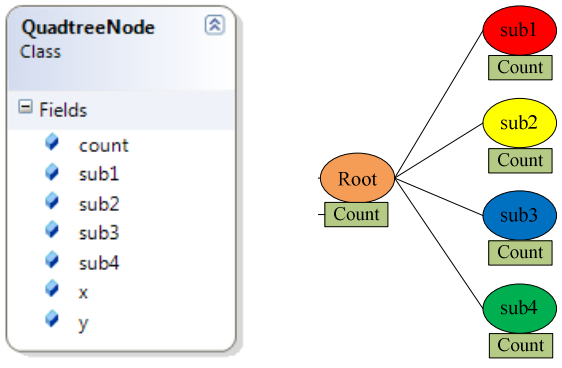
In order to compare the physical performance with and without using voice interaction in a solid modeling process, a mouse tracker is developed to monitor the mouse movement. When a solid modeling process finishes, the mouse positions are serialized to a data file, in the form of a coordinate list $L=\{(x_i, y_i)\}$, $1 \le i \le n$, where n is the total number of mouse locations. To answer the aforementioned questions, directly looping through all the points $\{(x_i, y_i)\}$ and then accumulating the count of mouse cursor occurrence at specific locations is, of course, a feasible and intuitive way; however the computation overhead is generally expensive, and the above query usually takes O(n) time. Using a quadtree structure, however, the same task can be accomplished in O (log n) time.

B. Quadtree construction

A quadtree is an efficient data structure to store and query spatially distributive data. It recursively participate the domain of interest into four quadrants and the data storage can be allocated as requested dynamically. Compared with the uniform image-like data arrays, the memory consumption is much lower.

A quadtree structure can be constructed by reading the data in the file and sequentially inserting a quadtree node each time until the entire coordinate list is populated. The structure of a quadtree node is illustrated in Fig. 1 and each node contains:

- A coordinate (x, y) indicating where the mouse cursor is located.
- An integer variable "count" showing how many times the mouse has been located at (x, y).
- And four *sub*-nodes, namely the *sub*1, *sub*2, *sub*3 and *sub*4 as shown in Fig.1 (b).



(a) (b)

Figure 1 The proposed quadtree node structure.

For a given node N, its *sub*-nodes are designed to satisfy the constraints shown in Fig. 2. For instance, the coordinates of $N.sub1$follows $N.sub1.x \ge N.x$ and $N.sub1.y \ge N.y$ and such rules are applicable to all the nodes and their *sub*-nodes.

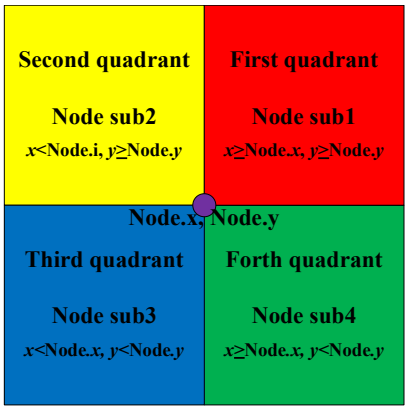


Figure 2 A quadtree node and its *sub*-nodes.

By the above definition, a quadtree can be used to partition a 2D space by recursively subdividing it into four quadrants or sub-regions. The construction of a quadtree is essentially based on the same principle as the building of a binary search tree [17]. Given the mouse coordinate list $L=\{(x_i, y_i)\}$, its quadtree representation can be constructed using the algorithm shown in Fig. 3.

```
Insert( Point (xi,yi), Node N )   //Insert the Point(xi,yi ) into Node N
{
        IF (Node N is null){
                N= new Node ( ) ;   N.x = xi;    N.y = yi;
        }
        ELSE
        {
                IF (x, y) =(N.x, N.y)                    //①
                        INCREMENT N.count;
                ELSE IF (xi ≥ N.x  && yi ≥ N.y)     //②
                {       //Insert the Point(xi,yi) into the first child of Node N
                        Insert(Point(xi,yi), N.sub1)
                }
                ELSE IF (xi  < N.x  && yi ≥ N.y)     //③
                {       //Insert the Point(xi,yi) into the second child of Node N
                        Insert(Point(xi,yi), N.sub2)
                }
                ELSE IF (xi < N.x  && yi < N.y)      //④
                {       //Insert the Point(xi,yi) into the third child of Node N
                        Insert(Point(xi,yi), N.sub3)
                }
                ELSE IF (xi ≥ N.x  && yi < N.y)      //⑤
                {       //Insert the Point(xi,yi) into the fourth child of Node n
                        Insert(Point(xi,yi ), N.sub4)
                }
        }
}
```

Figure 3 Pseudo code to construct a quadtree from a list of mouse coordinates.

Suppose the recorded mouse cursor locations are {A(5,5), B(7,6), C(2,8), D(4,3), E(1,7), F(3,8), G(2,1), H(8,2), I(3,9), J(1,9), K(4,9), L(5,5), M(2,8)}, the following steps are used to construct the quadtree:

1) Create a new quadtree, set the point A (5, 5) as the root node.

2) Insert point B into the *sub*1 node of A, as both the x and y coordinates of Point B are bigger than those of A.

3) Similarly insert Point C(2,8) into *sub*2 node of A, which corresponds to the case ③ in Fig 3.

4) Insert Point D(4,3) into *sub*3 Node of A, as the case ④ in Fig 3; insert Point D(8, 2) into *sub*4 Node of A, as the case ⑤ in Fig 3;

5) For Point E(1,7), it should be inserted into the branch rooted by the node C(2,8) as its coordinates and the root node's coordinates match the case ③ in the Fig. 3. Again, the above process is recursively called to conduct the second level comparison. As the *x* coordinate of Point E (which is 1) is smaller than that of Point C (which is 2), and the *y* coordinate of Point E (i.e. 7) is *smaller* than that of Point C (i.e. 8), therefore the case ④ in Fig. 3 is met, and the Point E(1,7) is then inserted into the third quadrant (*sub*3) of the Point C, as shown in Fig. 4.

6) Similarly the Point F to Point K are sequntially inserted into the quadtree.

7) As Point L is same as Point A, Point M is same as Point C, which indicates that the case ① in Fig. 3 is met, so instead of inserting new quadtree node, we simply increment the varible "count" in respective Nodes, therefore the count of node A and node C are both "2", as shown in Fig. 4.

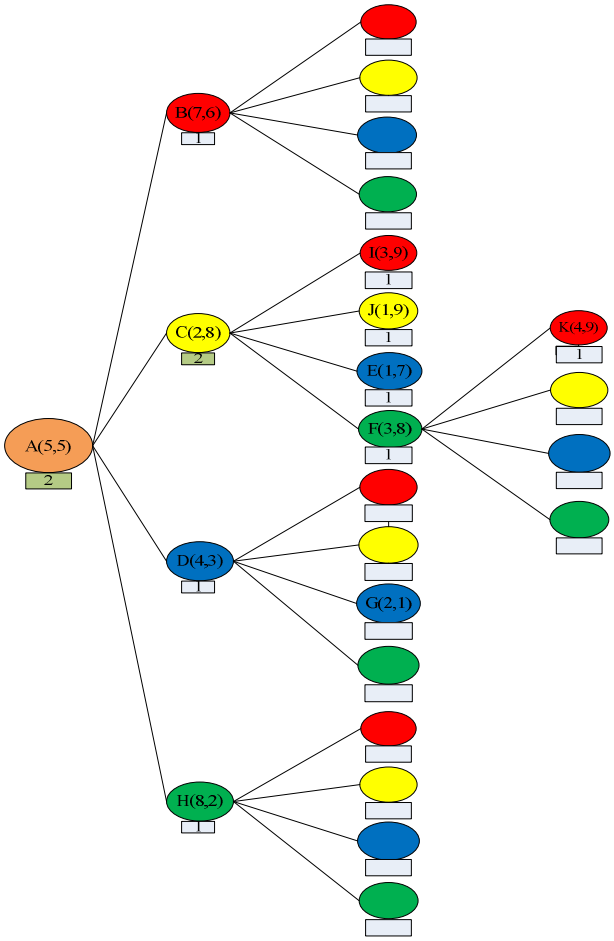


Figure 4 A quadtree representation for the list of cursor locations.

C. *Quadtree based mouse trajectory analysis*

In a general setting, all the problems raised in Section II *A* can be formulated into or amount to the following problem: given a quadtree representation of the mouse positions, get the count/cardinality of the point set *P* in a region/block *S*:

$$P = \{(x_i, y_i) | l \le x_i \le r, b \le y_i \le t\} \quad (1)$$

where *l* (left), *r* (right), *b* (bottom) and *t* (top) are the bounding range of block S. For instance, a typical query might be "How many points are there in the block bounded by $200 \le X \le 300$ and $100 \le Y \le 400$?"

To get the point count of the set *P*, we need to traverse the quadtree following specific logics. Denote the function to accomplish this as *search*(x_0, x_1, y_0, y_1, *N*), which returns the number of points in the tree branch root by the Node *N*, where x_0, x_1, y_0, and y_1 are the bounding limits of the block, and *N* is the quadtree node under interrogation. The quadtree traversal problem is essentially to get the output of the function *search*(*l*, *r*, *t*, *b*, Node), where Node stands for the root node of a quadtree.

Depending on the node's relative positions (Node.*x*, Node.*y*) with block *S*'s locations, there are nine possible configurations as illustrated in Fig. 5.

1) Both corrdinates of the Node are in the block *S*'s range, i.e. the Node is completely inside block *S*, as shown in the regions I in Fig. 5.

2) The Node is outside block *S*, as shown in the regions II, III, IV and V in Fig. 5.

3) Only one of the corrdinates of the Node, either *x* or *y*, is in the block *S*'s bounding range, for instance, the regions VI, VII, IX and XIII in Fig. 5.

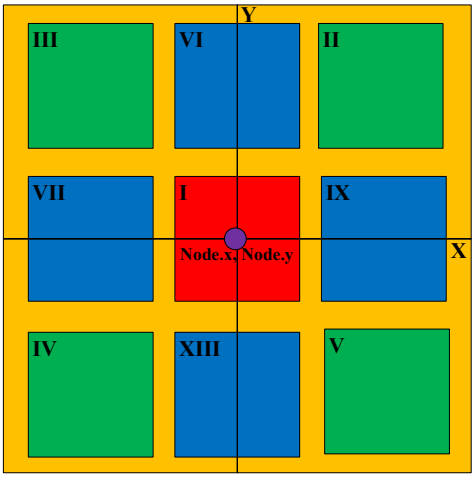


Figure 5 Nine possible cases of relative positions of a Node and a block.

Suppose the block *S* is in the location of case II in Fig. 5, and the root node is completely outside *S*, that is:

$$\text{Node}.x < l, \quad \text{Node}.y < b \quad (2)$$

According to the properties of the quadtree representation, it is also known that:

$$\text{Node.sub2}.x < \text{Node}.x$$
$$\text{Node.sub3}.x < \text{Node}.x \quad (3)$$
$$\text{Node.sub4}.y < \text{Node}.y$$

Combining (2) and (3), it is easy to see that the *sub*2, *sub*3 and *sub*4 nodes and all their children nodes are outside block *S* (because of violation of (1)). In other words, to count the points inside the block *S*, we can further confine the search within the branch rooted by ***sub*1** only, and there is no need to traverse the entire quadtree instead, as shown in case II in Tab. I. This significantly limits the searching space and saves the required computation resources. The merit of such a property is not only limited at the root node, and is in fact applicable to the nodes at all hierarchies. The same process can be repeated until the leaf nodes at specific branches are interrogated. Tab. I lists all the possible cases of recursive calls to the *search* function, which avoids brute-force quadtree node traversals.

TABLE I. RECURSIVE SEARCHING POINTS ROOTED BY A NODE

Case	Conditions	Actions

I	$l < x < r$
	$b < y < t$
		if (*n.sub*1 != null) *n.x*
		search(*n.x*, *r*, *t*, *n.y*, *n.sub*1);
		if (*n.sub*2 != null)
		search(*l*, *n.x*, *t*, *n.y*, *n.sub*2);
		if (*n.sub*3 != null)
		search(*l*, *n.x*, *n.y*, *b*, *n.sub*3);
		if (*n.sub*4 != null)
		search(*n.x*, *r*, *n.y*, *b*, *n.sub*4);

II	$r > x,\ l > x,\ b > y,\ t > y$	if (*n.sub*1 != null) search(*l*, *r*, *t*, *b*, *n.**sub*1**);
III	$r < x,\ l < x,\ b > y,\ t > y$	if (*n.sub*2 != null) search(*l*, *r*, *t*, *b*, *n.sub*2);
IV	$r < x,\ l < x,\ b < y, t < y$	if (*n.sub*3 != null) search(*l*, *r*, *t*, *b*, *n.sub*3);
V	$r > x,\ l > x,\ b < y, t < y$	if (*n.sub*4 != null) search(*l*, *r*, *t*, *b*, *n.sub*4);

VI	$r > x,\ l < x,\ b > y,\ t > y$	if (*n.sub*2 != null)
		search(*l*, *n.x*, *t*, *b*, *n.sub*2);
		if (*n.sub*1 != null)
		search(*n.x*, *r*, *t*, *b*, *n.sub*1);

VII	$r < x,\ l < x,\ b < y,\ t > y$	if (*n.sub*2 != null)
		search(*l*, *r*, *t*, *n.y*, *n.sub*2);
		if (*n.sub*3 != null)
		search(*l*, *r*, *n.y*, *b*, *n.sub*3);

VIII	$r > x,\ l < x,\ b < y,\ t < y$	if (*n.sub*4 != null)
		search(*n.x*, *r*, *t*, *b*, *n.sub*4);
		if (*n.sub*3 != null)
		search(*l*, *n.x*, *t*, *b*, *n.sub*3);

IX	$r > x,\ l > x,\ b < y,\ t > y$	if (*n.sub*1 != null)
		search(*l*, *r*, *t*, *n.y*, *n.sub*1);
		if (*n.sub*4 != null)
		search(*l*, *r*, *n.y*, *b*, *n.sub*4);

As special cases of the search within a block/region, if the query is concerned with only one of the coordinates, then the block search/query degenerates to a search/query within a bar-like region, and (1) turns to be (4) or (5):

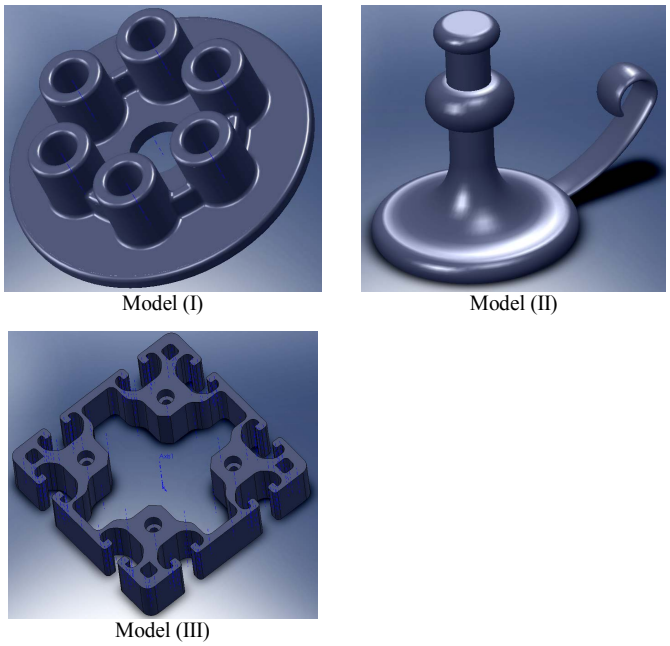
$$P = \{(x_i, y_i) \mid l \le x_i \le r\} \quad (4)$$

$$P = \{(x_i, y_i) \mid b \le y_i \le t\} \quad (5)$$

In this circumstance, the conditions to be checked in Tab. I are further reduced in a half, and the possible nine configurations also reduce to three.

III. CASE STUDIES

Based on the proposed approach discussed earlier, we have constructed three nontrivial solid models (from SolidWorks tutorials) using both the VoiceCAD and traditional CAD modelers, as shown in Fig. 7.



Model (I) Model (II) Model (III)

Figure 6 Three non-trivial solids modeled with and without using voice based interactions.

Two groups of experiments are conducted. The 1st query is to get the count of mouse locations which fall inside the menu and toolbar areas and the searching block is the region bounded by $0 \le x \le 1440$, $0 \le y \le 110$; the 2nd query provides the count of mouse occurrence in the "property manager page" ($0 \le x \le 190$, $125 \le y \le 775$), as shown in Fig. 8. These experiments are conducted on the PC with 2.5G memory and Intel 1.6GHz CPU, and the resolution of the screen is set to 1440 X 900.

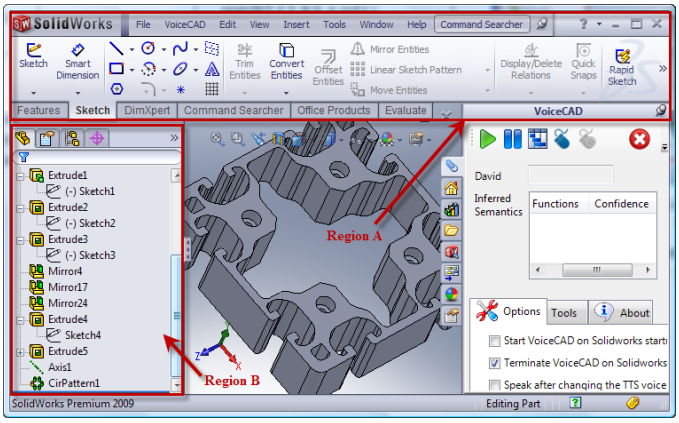


Figure 7 A snapshot of the VoiceCAD interface and the region of bar search and block search.

The above mouse trajectory analysis is separately conducted on the three models, and the statistics are listed in Tab. I and Tab II.

TABLE II. MOUSE OCCURANCE IN REGION A AND B, VOICE DISABLED

Models	Number of cursor locations recorded	Number of cursor locations in Region A	Number of cursor locations in Region B
(I)	16295	3336	3973
(II)	39197	5838	8852
(III)	82939	9907	13065

TABLE III. MOUSE OCCURANCE IN REGION A AND B, VOICE ENABLED

Models	Number of cursor locations recorded	Number of cursor locations in Region A	Number of cursor locations in Region B
(I)	8420	280	1270
(II)	19522	234	2538
(III)	35302	386	1532

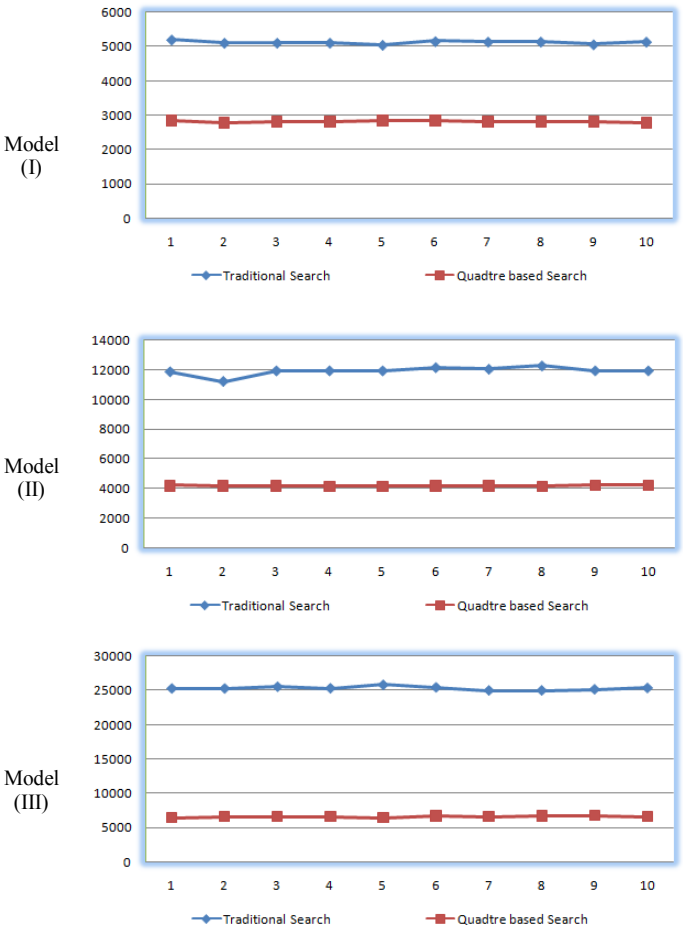


Figure 8 Time consumptions for mouse distribution statistics in the toolbar region A.

As is seen from the tables, the mouse movement is significantly reduced when voice is enabled to fire CAD modeling commands, with a total 48.3%, 50.2% and 57.4% reduction for model (I), (II) and (III) respectively.

When voice is disabled, 20.4%, 14.9% and 11.9% of the mouse movement is located in the toolbar area for the construction of the CAD model (I), (II) and (III), whereas these data are 3.3%, 1.2% and 1.1% when voice is enabled. There are still few mouse occurrences in the toolbar area as the speech recognition is not completely accurate, and several commands are activated with mouse clicks in exceptional cases.

In the property manage pane, 24.4%, 22.5% and 15.8% of the total mouse movements occur there for model (I), (II) and (III) when voice interaction is disabled, and when voice is enabled, these figures are 15.1%, 13% and 4.3%. There is no significant improvement in mouse movement saving in this region, as currently the modeling parameters are not fully configurable with voice commands, and mixed mouse/voice interactions are used.

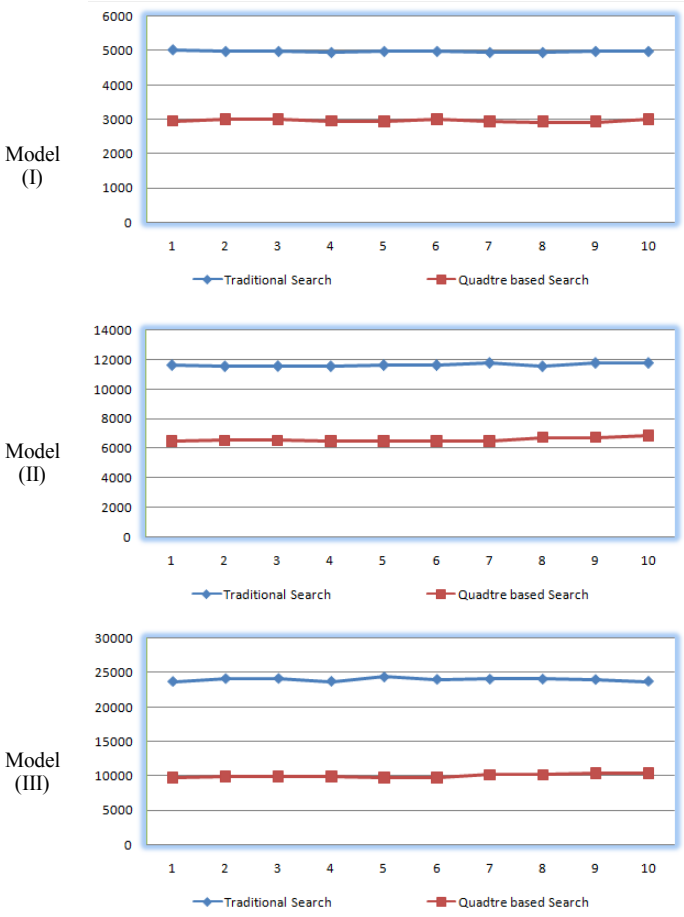


Figure 9 Time consumptions for mouse distribution statistics in the property manager region B.

The data in Tab. I and Tab. II are derived using the proposed quadtree based approach, and for comparison purposes, we also carried out the same analysis using the traditional brute-force approaches which require traversal of all the points in the data files. Fig. 8 and Fig. 9 illustrate the time consumptions to get these statistics in the region A and B respectively. The vertical axis is the total time consumption (in microseconds) of 10, 000 queries, and the same data queries are

repeatedly conducted ten times, as shown in the horizontal axis. It can be seen that the proposed approach is more efficient than the traditional approaches.

IV. Conclusions and Discussion

This paper proposes a quadtree based approach to analyze the mouse movement distributions in our VoiceCAD system. The mouse tracker keeps a record of all the mouse movement during the solid modeling process. A quadtree based approach is then applied to analyze the mouse trajectory distributions in both the traditional CAD and the VoiceCAD system. Our experiments show that the mouse movement is significantly reduced when voice command is used to activate CAD modeling commands.

Acknowledgment

The authors would like to thank the Department of Mechanical Engineering, the Committee on Research and Conference Grants of the University of Hong Kong and to the Research Grants Council of Hong Kong SAR government for supporting the projects.

References

[1] C. Becchetti, *Speech recognition : theory and C++ implementation*. Chichester :: Wiley, 1998.

[2] M. R. Schroeder, *Computer speech : recognition, compression, synthesis ; with introductions to hearing and signal analysis and a glossary of speech and computer terms*, 2nd ed. ed. Berlin ; New York :: Springer, 2004.

[3] J. N. Holmes, *Speech synthesis and recognition*, 2nd ed. ed. London ; New York :: Taylor & Francis, 2001.

[4] N. Diniz and C. Branco, "An approach to 3D digital design free hand form generation," in *Proceedings of the International Conference on Information Visualization*, London, 2004, pp. 239-244.

[5] A. G. Michael, S. E. David, and W. F. Timothy, *The integrality of speech in multimodal interfaces* vol. 5: ACM Press, 1998.

[6] R. D. Peacocke and D. H. Graf, "An Introduction to Speech and Speaker Recognition," *Computer,* vol. 23, pp. 26-33, 1990.

[7] T. H. Dani and R. Gadh, "COVIRDS: a conceptual virtual design system," in *ASME Database Symposium*, Boston, MA, USA, 1995, pp. 959-966.

[8] T. H. Dani and R. Gadh, "Creation of concept shape designs via a virtual reality interface," *CAD Computer Aided Design,* vol. 29, pp. 555-563, 1997.

[9] S. Gao, H. Wan, and Q. Peng, "An approach to solid modeling in a semi-immersive virtual environment," *Computers & Graphics,* vol. 24, pp. 191-202, 2000.

[10] C. P. P. Chu, T. H. Dani, and R. Gadh, "Multi-sensory user interface for a virtual-reality-based computer-aided design system," *CAD Computer Aided Design,* vol. 29, pp. 709-725, 1997.

[11] M. Weyrich and P. Drews, "An interactive environment for virtual manufacturing: The virtual workbench," *Computers in Industry,* vol. 38, pp. 5-15, 1999.

[12] R. A. Bolt and E. Herranz, "Two-handed gesture in multi-modal natural dialog," in *Proceedings of the 5th annual ACM symposium on User interface software and technology* Monteray, California, United States: ACM, 1992.

[13] X. Y. Kou and S. T. Tan, "Design by talking with computers," *Computer-Aided Design and Applications,* vol. 5(1-4), pp. 266-277, 2008.

[14] S. Xue, X. Y. Kou, and S. T. Tan, "Natrual Voice-enabled CAD: Modelling via Natural Discource," in *the 2009 International CAD Conference* Reno, Nevada, June 2009, Accepted.

[15] Speak4CAD, http://www.speak4cad.com/

[16] ThinkDesign, http://www.think3.com/en/default.aspx.

[17] D. P. Mehta and S. Sahni, *Handbook of data structures and applications*. Boca Raton, Fla. :: Chapman & Hall/CRC, 2005.