THE INTERNATIONAL JOURNAL OF MEDICAL ROBOTICS AND COMPUTER ASSISTED SURGERY
Int J Med Robotics Comput Assist Surg 2008; **4**: 77–86.
Published online 14 February 2008 in Wiley InterScience (www.interscience.wiley.com). **DOI:** 10.1002/rcs.177

ORIGINAL ARTICLE

Hierarchical spatial hashing-based collision detection and hybrid collision response in a haptic surgery simulator

X. Li[1]
L. Gu[1,2]*
S. Zhang[2]
J. Zhang[2]
G. Zheng[2]
P. Huang[2]
J. Xu[3]*

[1]*Computer Science and Engineering Department, Shanghai Jiao Tong University, Shanghai, People's Republic of China*

[2]*School of Software, Shanghai Jiao Tong University, Shanghai, People's Republic of China*

[3]*Shanghai Renji Hospital, Shanghai, People's Republic of China*

*Correspondence to: L. Gu, Computer Science and Engineering Department, Shanghai Jiao Tong University, Shanghai, People's Republic of China.
E-mail: gu-lx@cs.sjtu.edu.cn or J. Xu, Shanghai Renji Hospital, Shanghai, People's Republic of China, E-mail: xujiaar@online.sh.ca

Accepted: 21 December 2007

Abstract

Background Collision detection and response are two crucial aspects in a virtual surgery simulator, which significantly affect the output in real-time response and simulation realism.

Methods We propose a collision detection algorithm which employs a novel hierarchical spatial hashing to effectively achieve appropriate parameters. Thereafter we present a hybrid collision response scheme which takes the advantages of three traditional methods and introduces a 'force filter' to obtain more realistic feedback force.

Results The average performance of our proposed collision detection has been improved by 15.65% against the traditional method. After collision response processing the smoothness of feedback force has been greatly enhanced, and this indicates a more realistic feedback force without restricting the range of force inputs.

Conclusions The experimental results reveal that the proposed approaches can achieve the real-time response and simulation realism required by a haptic surgery simulator. Copyright © 2008 John Wiley & Sons, Ltd.

Keywords virtual surgery; collision detection; collision response; feedback force

Introduction

Background

Minimally invasive surgery (MIS) has become increasingly important in recent years. With the development of the virtual reality (VR) technology and haptic devices, many VR-based systems presenting comprehensive virtual organ models appear to be widely in use by surgeons in training for the purpose of repeatedly practising surgical procedures. However, the operation is too complicated to manipulate, since it is difficult for surgeons to know the exact depth of a surgical tool on a two-dimensional screen. After careful discussions with surgeons, we found that both vision and haptic feedback are necessary; we need to know when and where the tool touches an organ, how the organ deforms and how much the feedback force is. Collision detection determines when and where two geometrical entities touch each other, and collision response computes the behaviour after the collision has been detected. Therefore, collision detection and collision response algorithms are extremely important in implementing a

haptic surgery simulator (HSS). However, implementing an effective and stable collision detection algorithm in a real-time response required system and finding a collision response method to generate some reasonable feedback force in a haptic required system constitute a considerable challenge.

We are developing a haptic laparoscopic surgery simulator, whose current target is to incise a tumour on a kidney in the virtual scene. In order to obtain real-time response and simulation realism, we proposed an improved collision detection algorithm for detecting the collision between deformable objects, and a hybrid collision response scheme for calculating deformation of soft tissues as well as generating continuous feedback force. These two approaches are designed to be general methods for most surgical simulators which require collision detection between deformable models and/or calculation of a realistic feedback force. This paper focuses on the two important methods invented and implemented in an HSS.

Related work

Collision detection

The aim of collision detection is to determine intersections between geometrical entities. In a HSS, the interactions to be detected are usually interactions of some surgical tools with soft tissues, and those soft tissues are always deformable models. Many algorithms for detecting collision between deformable models have been proposed in recent years (1–3); however, in our case, the intersections to be detected were between tetrahedra and vertices, whereas the methods above used different models. There are two traditional approaches can be used in our case: spatial partitioning and bounding volume hierarchies. Early work in this area has been done: in (4), uniform subdivision spatial hashing was used to interactively detect collisions between molecules; (5) presented a hierarchical spatial hashing scheme for collision detection in robot motion planning, but this approach was limited to collision detection between swept-volume bounding boxes in a rigid body environment; in (6), a uniform subdivision spatial hashing scheme was applied to efficiently detect collisions between deformable tetrahedral meshes. This approach performs very well; however, it has some limitations in choosing an appropriate grid resolution. We proposed an improved hierarchical spatial hashing scheme (7) for overcoming this limitation, which is described below.

Collision response

In an HSS, collision response computes the forces resulting from a collision, which can be categorized into two classes, the force causing soft body's deformation and the feedback force. Collision response schemes suitable for real-time deformable model handling were mainly categorized into three distinct approaches, called the penalty force approach, the contact surface computation and the analytical solution. The idea of a penalty force is first mentioned in (8). Penalty-based methods evolved to constraint-based methods proposed in (9). This method has good efficiency, although influenced by penetration. Contact force-based algorithms have also been discussed (10–15); this scheme avoids penetration but has a high time consumption. The analytical collision response approach applied to deformable models is discussed in (16–18); it is very precise and also avoids penetration, but it also causes an extreme computational overhead.

Another important task of HSS is to provide a feedback force approximation to the real operation. A variety of forces for haptic simulation has been presented in (19); (20) presents a method for creating surgical simulators with high-fidelity haptic feedback. A reasonable feedback force should continuously be kept on scale and direction in order to eliminate an unnatural feeling for the operators.

After analysing the advantages and disadvantages of existing methods, a hybrid collision response method was initially proposed in (21), and it has been systemized and a 'force filter' also introduced in this study to obtain a more realistic feedback force.

Materials and methods

System overview

We developed an HSS for laparoscope-based MIS training. It integrated a set of functions, including a mathematical model for soft tissue deformation and collision detection as well as response and cutting, where volume meshing is the fundamental procedure for the simulation. Figure 1 shows the framework of the whole system, and we mainly focus on the implement of collision detection and collision response in this paper.

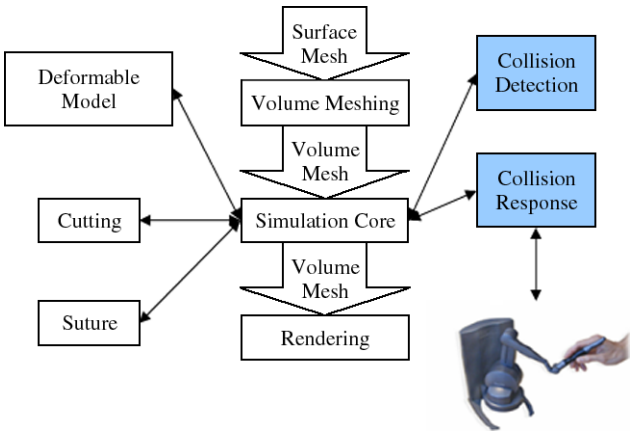

Figure 1. The simulator framework

DOI: 10.1002/rcs

Collision detection

Hierarchical subdivision

The method subdivides space into a hierarchical grid, which is made up of axis-aligned cubes, the so-called grid cells, with a flexible edge-length k, whose edge length is called the grid cell size.

Assume that a scene of n objects consists totally of m tetrahedra. In a first pass, for each of the m tetrahedrons, it determines which grid cells the tetrahedron occupies, and the grid cell size is optimized for each tetrahedron. Let $s = size(t)$ be the length of the longest edge of the axis-aligned bounding box of a tetrahedron t; the grid cell size k which is to be used for the mapping of t is then defined as:

$$k = 2^{[\log_2^{(s)}]} \tag{1}$$

According to equation 1, in a one-dimensional case an object never occupies more than two grid cells, and less than four or eight in a two- or three-dimensional case, respectively. Here $[\log_2^{(s)}]$ defines the spatial grid's subdivision level, named l, and it determines how coarsely the spatial grid is subdivided.

Mapping

Since space is subdivided into a hierarchy of grid cells, each point (x, y, z) belongs to a unique grid cell. The aim of mapping point (x, y, z) to the address of the grid cell in which it is contained is to find a unique *address* for each possible grid cell. The mapping is defined as follows:

$$\begin{bmatrix} x \\ y \\ z \end{bmatrix} = \begin{bmatrix} [x/k] \\ [y/k] \\ [z/k] \\ l \end{bmatrix} \tag{2}$$

where k is the grid cell size and l is the subdivision level. Note that smaller grid cells contained in a bigger one also need a unique *address*. Then a hash function is chosen to map the address (x, y, z, l) uniformly over the whole hash table. It is important that spatial nearness does not result in nearness in the hash table. The hash function is defined as:

$$hash(x, y, z, l) = (xp_1 \oplus yp_2 \oplus zp_3 \oplus lp_4) \mod s_H \tag{3}$$

where p_1, p_2, p_3 and p_4 are large prime numbers, S_H is the size of the hash table and $\oplus$ denotes the XOR operator.

Point in tetrahedron test

After the mapping phase described above, each vertex of a tetrahedron is mapped into a grid cell, whose unique address is then mapped into a different entry of a hash table. Thus, each tetrahedron corresponds to several hash entries. Then, in the collision detection phase, the algorithm scans the hash table and takes only linear time to find the potential collisions between tetrahedra whose vertices occupy the same entry of the hash table. Finally, narrow-phase collision detection is employed in these tetrahedra which potentially collide, i.e. to test whether a point p is inside a tetrahedron t. There are multiple methods for such a test, e.g. to use barycentric coordinates to test whether a point p is inside a tetrahedron t [see (6) for more details]. Note that the partitioning process should be repeated and the hash table should be updated for every time step, since the objects will change positions during the simulation.

Collision response

Overview

The method divides the whole collision response procedure into three steps, the preprocessing step, the deformation step and the force feedback step (Figure 2). In an HSS, collision response has two tasks: one is to work out the force causing soft tissue deformation, which is implemented in the deformation step; another is to generate a realistic feedback force in the feedback step. In order to increase the stability and accuracy of the computation, another preprocessing step was added to refine the penetration depth.

Preprocessing step

In the preprocessing step, a kind of analytical solution approach was employed. We added backtracking steps at the time when collision was being detected, which led to a limited effect on the whole time complexity but increased the frequency of computation. The backtracking method is described as algorithm I, where *step* is the time of backtracking, *stepAmount* is a predefined threshold of step, *timeFactor* is a coefficient of time step, *epsilon* is a predefined small number, and when $timeFactor \leq epsilon$, or $step \geq stepAmount$ the process will be stopped. *Inside* is a coefficient that marks whether a point is inside an object, *currentPos* is the coordination of current point and *moveDir* is the vector of movement.

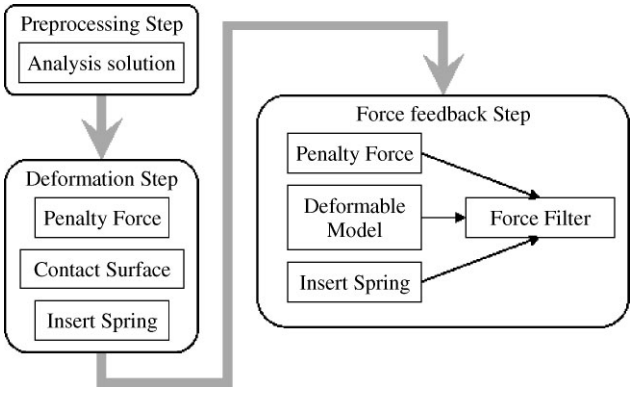

Figure 2. Three steps of the hybrid collision response method

DOI: 10.1002/rcs

Algorithm I

```
begin
    Step ← 1
    timeFactor ← 1
    while step < stepAmount and
      timeFactor > epsilon do
        if Inside (point) then
              inside ← 1
        else
              inside ← −1
        end if
              timeFactor ← 1/2^step
              currentPos ← currentPos
              + inside * timeFactor
               * moveDir
              step← step+1
        end while
              return currentPos
end
```

Deformation step

In the deformation step, the optional algorithms are the penalty force-based algorithm (22), the contact force-based algorithm (23) and the spring model-based algorithm, to which are added some springs between two objects when they are colliding (Figure 3). The nodes connected to springs will finally return to the balance position, and the collision force between these two objects will be released.

The choice of algorithm should depend on the requirements of different systems. The penalty force-based and spring model-based algorithms are suitable for a real-time response primary system, while the contact force-based algorithm is appropriate in a system where a high quality of visual effect is required. The spring model-based algorithm is also suitable for a system which requires high stability.

Force feedback step

In the force feedback step there are also three options: a penalty force-based method (22); a deformable model-based method, which calculates the internal force of each mass point and accumulates them to get the total response force of the collision area; and a spring model-based method. The deformable model-based method is more suitable for a real-time response-required system, whereas the spring model-based method is good for a force continuity-required system.

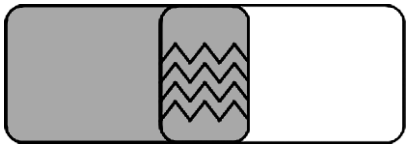

Figure 3. Spring model-based approach. When the grey entity collides with the white one, springs are inserted between them

Once the response force is calculated, a 'force filter' is introduced for the purpose of achieving a more realistic feedback force. A PHANToM Desktop (Sensable Inc., USA) is used in our HSS, having a limitation of output force of 0–7.9 N. Unfortunately, the response force we calculated is far out of this range, and the 'force filter' is used to solve this problem without restriction of input scope, which includes two steps: mapping and smoothing.

Equation (4) (see Appendix I for more details) is used to map force into a smaller range in the mapping step. Here *forceX, forceY* and *forceZ* are response forces in three dimensions, *scaledForceX, scaledForceY* and *scaledForceZ* are the respective forces after mapping, *forceThreshold* is the biggest force sent to the PHANToM Desktop, *forceMax* represents the biggest absolute value of the forces in three dimensions and *commonForce* is the force that appears with a high frequency:

$$\begin{cases} scaledForceX = \dfrac{2forceX \cdot \arctan[(forceMax \cdot \pi)/(4commonForce)]forceThreshold}{\pi \cdot forceMax} \\ scaledForceY = \dfrac{2forceY \cdot \arctan[(forceMax \cdot \pi)/(4commonForce)]forceThreshold}{\pi \cdot forceMax} \\ scaledForceZ = \dfrac{2forceZ \cdot \arctan[(forceMax \cdot \pi)/(4commonForce)]forceThreshold}{\pi \cdot forceMax} \end{cases} \quad (4)$$

In the smoothing step, a linear interpolation method is employed to smooth the force output to increase its quality of continuity.

Computation optimization

For the real-time capability, the visual display required an update rate of 30 Hz, while a stable haptic display required an update rate of 1000 Hz. Due to the large gap existing between these two requirements and our willingness to take advantage of the widely used dual-core processor, it was desirable to separate the device-control module from the whole simulator. A server was designed to control all haptic feedback devices, which communicated with the simulator through inter-process communication (IPC) methods (Figure 4). Because the loop of the server was running much higher than that of the client, information from the device, such as position, were not sent unless a request from the client was received. This design was able to increase the overall performance of the dual-core-based computer, since the server and client ran on two processors. It also contributed to maintainability and adaptability.

Results

System development

Our laparoscope-based HSS (Figure 5) was developed on a platform with Pentium IV 2.6 GHz CPU, 2 GB RAM and a

DOI: 10.1002/rcs

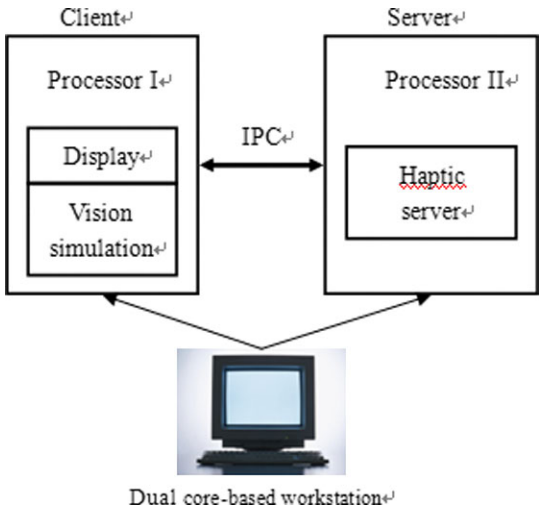

Figure 4. The dual core-based platform

GeForce 6800 graphic card. All the time critical simulation algorithms, such as collision detection and response, were developed in C++, and the necessary user interface was built using Python.

Collision detection

Here we compared the improved hierarchical spatial hashing scheme to the traditional spatial hashing algorithm, using a regular grid proposed by (6). Two objects were employed in the collision detection system, one sphere and one tetrahedral model derived from clinical kidney data. This configuration resulted in 524 vertices of the sphere that penetrated tetrahedrons of the kidney. Both objects together consisted of 49 848 tetrahedrons. Figure 6 shows a comparison of the performance of hierarchical spatial hashing against regular grid spatial hashing. Table 1 shows a performance comparison between hierarchical spatial hashing and regular spatial hashing, where the performance increment was calculated using equation (5), in which P is the performance increment, R is the run time of the regular grid method and H is the run time of the hierarchical grid method:

$$P = \frac{R - H}{R} \tag{5}$$

Table 1. Performance comparison between hierarchical spatial hashing and regular spatial hashing (7)

Scene	A	B	C	D	Average
Objects	2	100	36	9	–
Tetrahedra	49082	16200	1728	432	–
Vertices	14120	6400	972	243	–
Hierarchical grid (ms)	174	54	5.4	1.3	–
Regular grid (ms)	220	65	6.1	1.5	–
Performance increment (%)	20.9	16.9	11.5	13.3	15.65

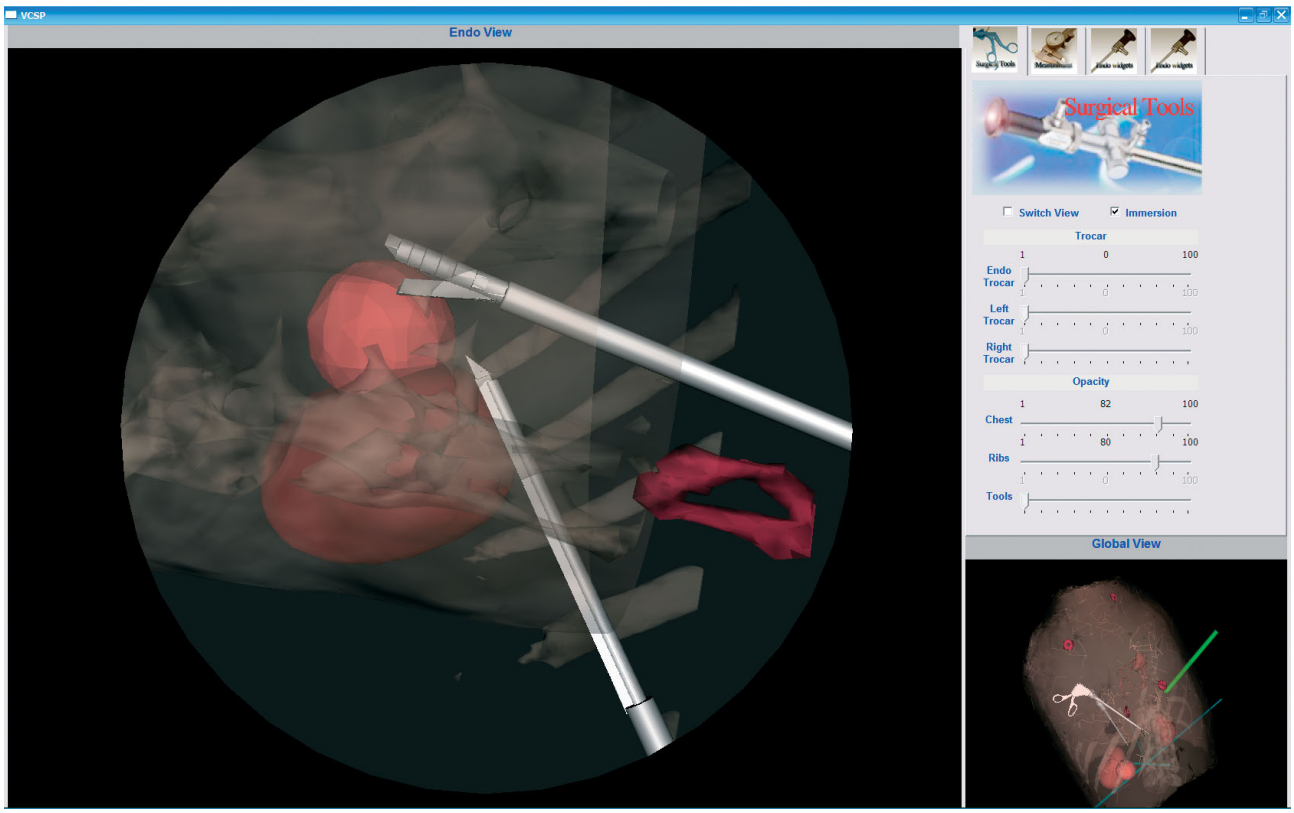

Figure 5. The snapshot of the system user interface

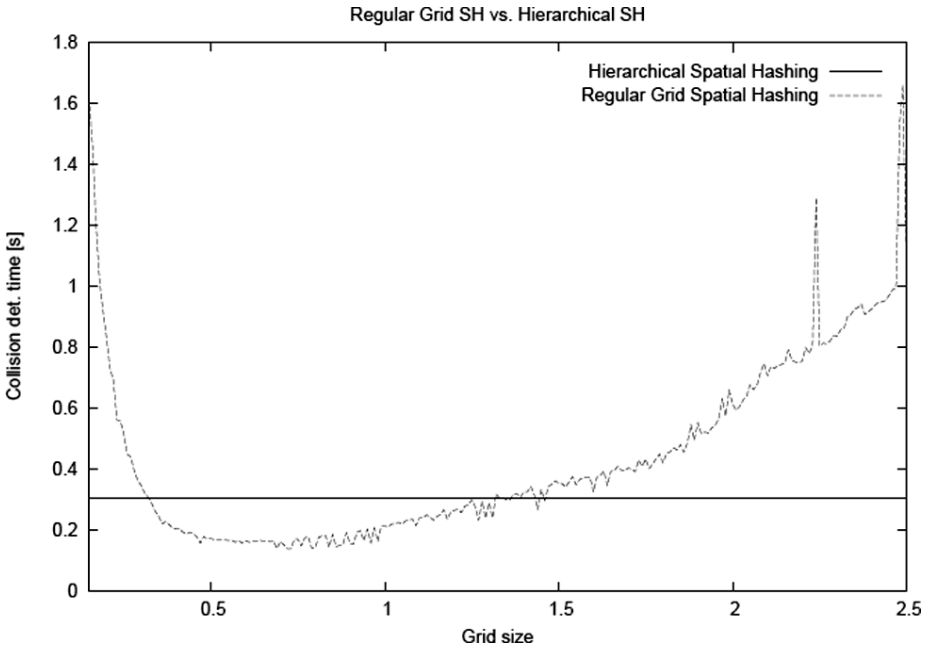

Figure 6. Performance comparison of hierarchical spatial hashing against regular grid spatial hashing

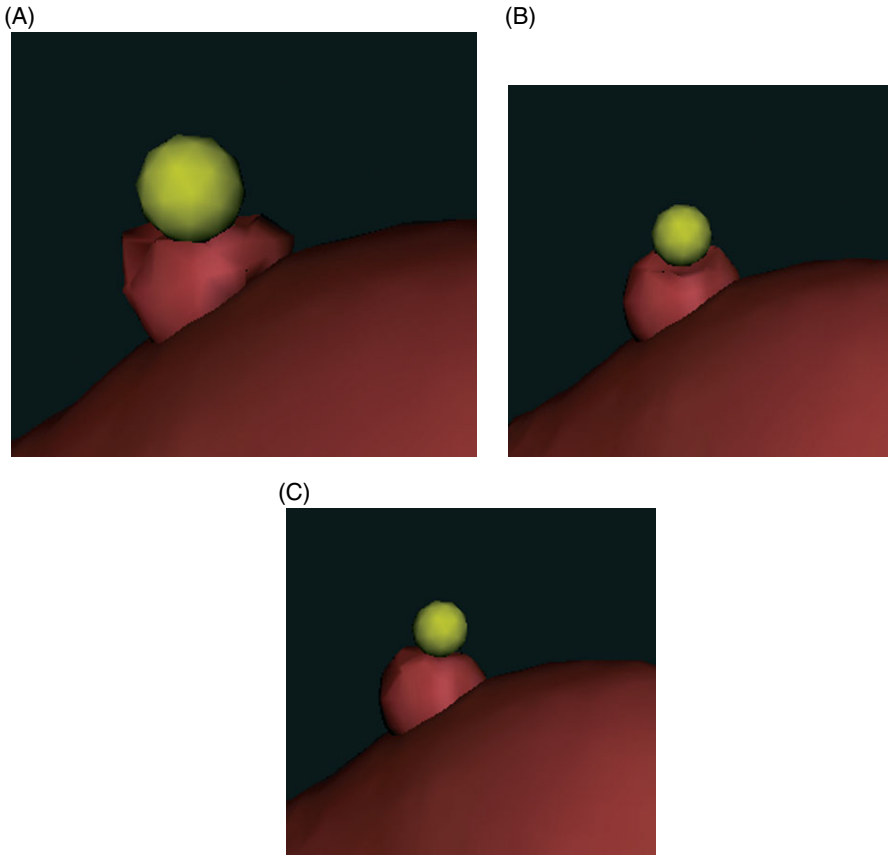

Figure 7. Comparison of deformations. The yellow sphere collides with a simulated tumour on a kidney. The tumour is deforming when collision has been detected, and a generated feedback force can be felt through the PHANToM. Deformation using: (a) penalty force-based algorithm; (b) contact force-based algorithm; (c) spring model-based algorithm

Collision response

Since the collision response algorithm had two tasks, the collision response experiment was divided into deformation evaluation and force feedback evaluation. In both we used clinical kidney data containing 1922 points and 6538 tetrahedra, a simulated tumour containing 206 points and 583 tetrahedra, and a sphere-shaped

DOI: 10.1002/rcs

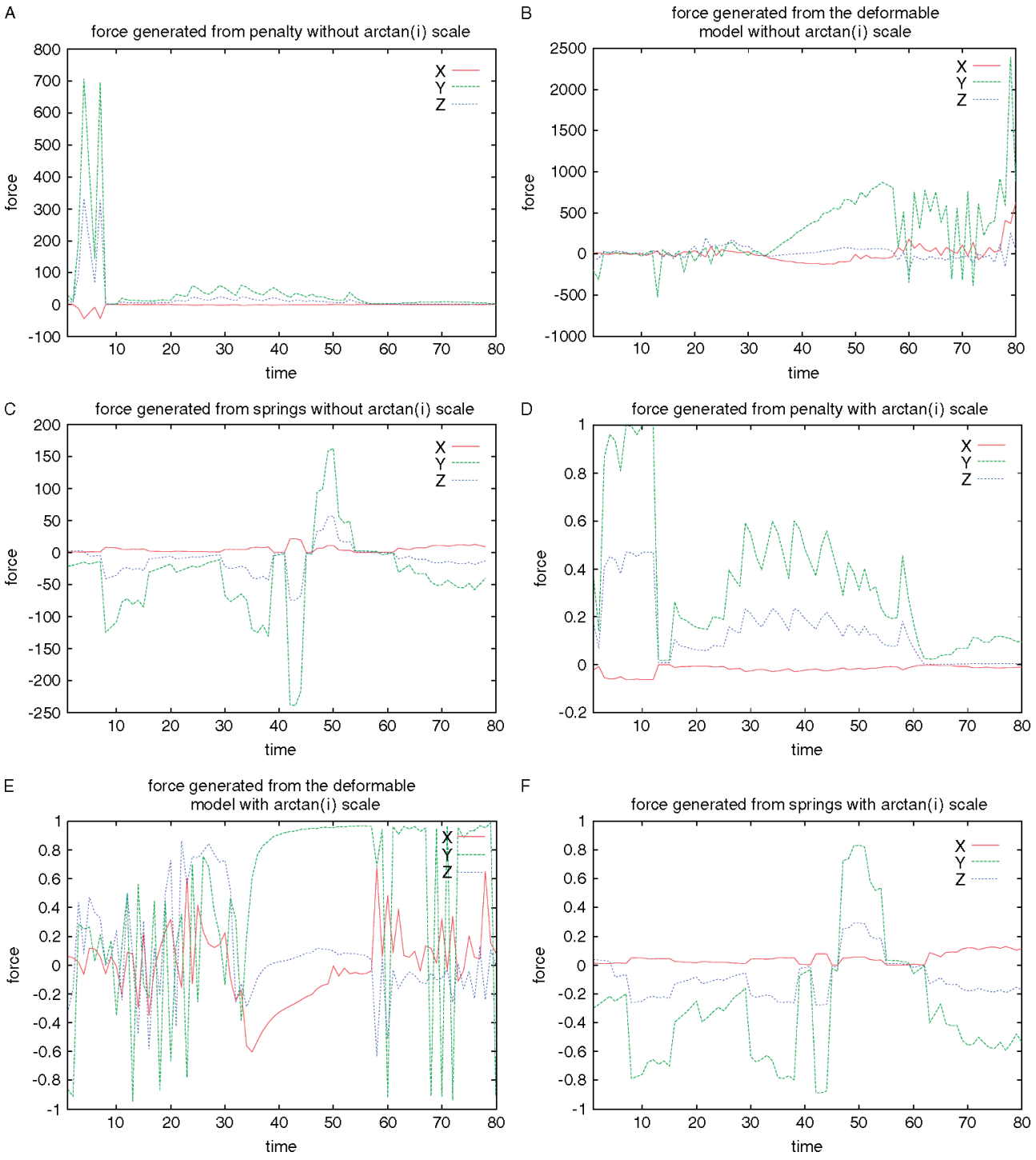

Figure 8. The comparison of forces continuity along *x*, *y* and *z* coordinates. Force generated from: (a) penalty force without arctan(i) scale; (b) deformable model without arctan(i) scale; (c) spring model without arctan(i) scale; (d) penalty force with arctan(i) scale; (e) deformable model with arctan(i) scale; (f) spring model with arctan(i) scale

tool containing 54 points and 164 tetrahedra. Figure 7 shows a comparison of the deformation using the penalty force-based, contact force-based and spring model-based methods, respectively.

The desired feedback force should be continuous and smooth; therefore we evaluated both the continuity and smoothness of the forces. First, we compared the continuity of forces calculated using the three methods mentioned above. Figure 8A–C shows the forces without the arctan(i) scale. Because a small fluctuation cannot be shown clearly in a compressed range, we needed to know the forces' condition after the arctan(i) scale (Figure 8D–F). Second, we evaluated the smoothness of the feedback force. Figure 9 shows a comparison between the forces before and after smoothing.

DOI: 10.1002/rcs

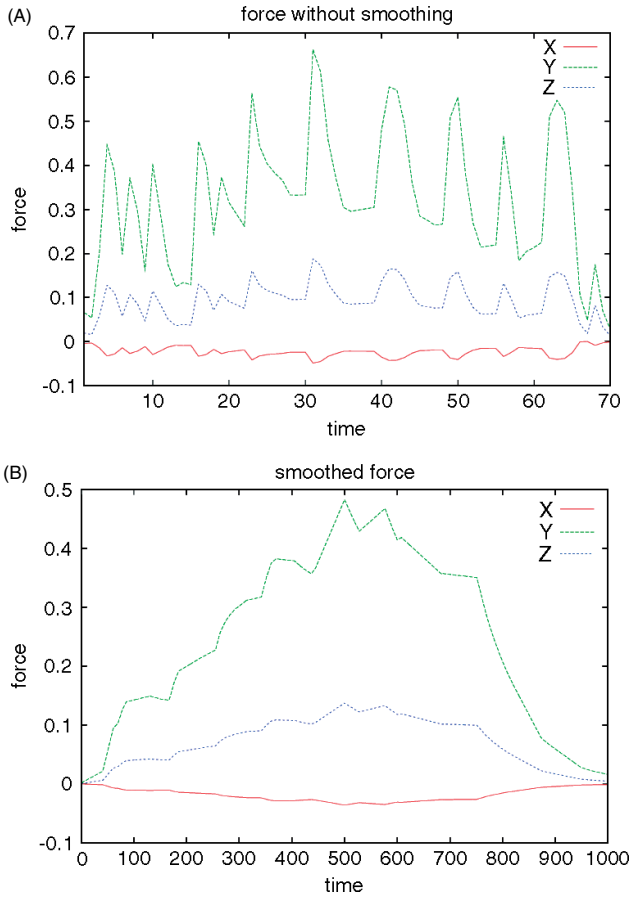

Figure 9. Comparison between forces in *x*, *y* and *z* coordinates, (a) before and (b) after smoothing

Discussion

Collision detection

As described above, the uniform grid approach (6) has some limitations in choosing an appropriate grid resolution. If the grid resolution is too high, one primitive occupies a large number of grid cells and the mapping process becomes extremely costly. In contrast, if the grid resolution is too low, many primitives are mapped into one single grid cell, which leads to high costs in the narrow phase, since many primitives have to be checked for intersection. Moreover, it is not easy to determine the recommended optimal grid size if the objects to be tested are made up of non-uniform-sized tetrahedra. The limitations above were overcome by using a hierarchical spatial hashing scheme here, and an optimal grid cell size for each tetrahedron could be achieved.

Table 1 shows that, compared with regular spatial hashing, the average performance of hierarchical spatial hashing was increased 15.65%. Especially in a case where the grid size was quite irregular (e.g. case A in Table 1), our method could significantly improve the performance (20.9%). This feature can also be found in Figure 6, where the proposed hierarchical spatial hashing is not dependent on user-defined parameters but is optimized automatically.

Collision response

The penalty force-based approach (8,9) is widely used in interactive real-time simulations, due to its computational efficiency. However, one of the biggest drawbacks of this method is penetration, which should be avoided in an HSS. The biggest advantage of the contact force computation approach (10–15) is that it avoids penetration; however, it needs a high sampling frequency, which cannot meet the requirements of a real-time response in an HSS. The advantages of the analytical collision response approach (16–18) are the high precision and lack of penetrations, but it will cause an extreme computational overhead, which is intolerable for a real-time response-required HSS. Our proposed hybrid approach uses the advantages of the three traditional approaches as well as eliminating their disadvantages. Modules in the three steps discussed above have high cohesion and low coupling, so there are many combinations, and which one will be selected depends on the requirements of each different system.

In the deformation step, if the penalty force-based algorithm is chosen, there is a problem in choosing an appropriate parameter. This problem also exists if we choose the contact surface based algorithm. However, both have a good visual effect (Figure 7A, B). Although the deformation is not so good if we choose a spring model-based algorithm (Figure 7C), this algorithm is much more stable if we set an appropriate elastic coefficient. This result is the same as discussed above.

In the force feedback step, as seen from Figure 8A–C, compared to the other two methods the penalty force-based method seems to generate the most continuous force. However, different from Figure 8A, the force is not so continuous, as shown in Figure 8D after the force in Figure 8A is mapped to (−1, 1) using equation (4). This is because the range has been compressed from (−1000, 2500) to (−1, 1); the small fluctuation which cannot be shown in Figure 8A has been magnified and is shown clearly in Figure 8D. On the other hand, the continuity of the forces generated from the spring model-based method, as shown in Figure 8C, F, are nearly the same. So in fact, the spring model-based method generates force with the best continuity, as discussed above. Moreover, from Figure 9 we find that the smoothness of force is greatly enhanced, which indicates that it better represents a more realistic force.

Clinical values

Advised by the surgeons from Shanghai Renji Hospital, our HSS is required to have a high quality of visual and haptic simulation effects as well as real-time ability. Therefore, the contact force-based algorithm was chosen in the second step of the collision response approach and the spring model-based method was used in the third step, and the output collision force was mapped to 0–2 N as the final feedback force.

DOI: 10.1002/rcs

By training on this system, laparoscopic procedure skills, such as hand–eye coordination, the fulcrum effect and depth perception, can be brought up as in the real operating room. This procedural skills training enables integration of knowledge and judgement into the technical skills already learned. This may lead to a reduction in the number of unnecessary complications occurring due to a failure of technical skills, and the time and expense spent acquiring basic laparoscopic skills in the operating room. This makes it possible to chart the performance of a trainee surgeon through the curriculum and define the attainment of proficiency.

Conclusion

In this paper, a hierarchical spatial hashing algorithm for a deformable model's collision detection is proposed which improves performance compared with a regular grid-based spatial hashing in non-uniform-sized tetrahedra. Meanwhile, a hybrid collision response method is also proposed. After the 'force filter', a continuous authentic feedback force could be generated. An HSS for laparoscope-based MIS training was developed based on these novel methods.

More clinical trials are intended to be implemented in future research to further fine-adjust the parameters by comparison to real surgical objects. We consider that, in the next step of practising this system, performance can be measured using parameters such as time taken, number of errors made and path length for each hand. In this way, the clinical value of the HSS system should be further validated.

Acknowledgements

We would like to express our gratitude to Dr Junfeng Cai from Shanghai Zhongshan Hospital and surgeons from Shanghai Renji Hospital for providing clinical data and advice. We are also grateful to our team members, Eitz Mathias, Jalda Dworzak and Jan Boehm, for providing helpful comments and hints. This research was partially supported by National 863 Research Fund of China (2007AA01Z312).

References

1. Sud A, Govindaraju N, Gayle R, Kabul I, and Manocha D. Fast proximity computation among deformable models using discrete Voronoi diagrams. *ACM Transactions on Graphics*, **25**(3): 1144–1153, July 2006.
2. Govindaraju NK, Knott D, Jain N, Kabul I, Tamstorf R, Gayle R, Lin MC, Manocha D. Interactive collision detection between deformable models using chromatic decomposition. International Conference on Computer Graphics and Interactive Techniques 2005, Dunedin, New Zealand, 991–999.
3. Galoppo N, Otaduy MA, Mecklenburg P, *et al*. Accelerated proximity queries for haptic rendering of deformable models. World Haptics Conference, Tsukuba, Japan, 2007.
4. Turk G. Interactive Collision Detection for Molecular Graphics. PhD Thesis, University of North Carolina, 1989.
5. Mirtich B. Efficient algorithms for two-phase collision detection. In *Practical Motion Planning in Robotics: Current Approaches and Future*, edited by K. Gupta and A.P. del Pobil, Cambridge University Press New York, NY, USA 1998; 203–223.
6. Teschner M, Heidelberger B, Mueller M, *et al*. Optimized spatial hashing for collision detection of deformable objects. *Proc Vision Model Visualiz* 2003; 47–54.
7. Mathias E, Gu LX. Hierarchical spatial hashing for real-time collision detection. Shape Modeling and Applications 2007; IEEE International Conference, June 2007; 61–70.
8. Terzopoulos D, Platt J, Barr A, *et al*. Elastically deformable models. *ACM SIGGRAPH Comput Graphics* 1987; **21**(4): 205–214.
9. Ruspini DC, Kolarov K, Khatib O. The haptic display of complex graphical environments. International Conference on Computer Graphics and Interactive Techniques, New York, NY, USA, 1997; 345–352.
10. Spillmann J, Teschner M. Contact surface computation for coarsely sampled deformable objects. Proc Vision Model Visualization 2005, Erlangen, Germany, 289–296.
11. Barbagli F, Prattichizzo D, Salisbury K. A multi-rate approach to haptic interaction with deformable objects: single and multipoint contacts. *The International Journal of Robotics Research* 2005; **24**(9): 703–715.
12. Balaniuk R. A differential method for the haptic rendering of deformable objects. Proceedings of the ACM Symposium on Virtual Reality Software and Technology 2006, Limassol, Cyprus, 297–304.
13. Duriez C, Andriot C, Kheddar A. Signorini's contact model for deformable objects in haptic simulations. IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), CEA, France, 2004; 4, 3232–3237.
14. Duriez C, Dubois F, Kheddar A, *et al*. Realistic haptic rendering of interacting deformable objects in virtual environments. *IEEE Trans Visualiz Comput Graphics* 2006; **12**: 364–367.
15. Pauly M, Pai DK, Guibas LJ. Quasi-rigid objects in contact. Proceedings of Symposium on Computer Animation, Grenoble, France, 2004; 1019–1091.
16. Baraff D. Analytical methods for dynamic simulation of non-penetrating rigid bodies. *ACM SIGGRAPH Comput Graphics* 1989; **23**(3): 223–232.
17. Johnson KL. Contact Mechanics. The Press Syndicate of the University of Cambridge, UK, 1985.
18. Oden JT, Kikuchi N. *Contact Problems in Elasticity: A Study of Variational Inequalities and Finite Element Methods*. Society for Industrial Mathematics, SIAM, Philadelphia, 1988.
19. http://www.sensable.com. 2005.
20. Mahvash M, Hayward V. High-fidelity haptic synthesis of contact with deformable bodies. *IEEE Comput Graphics Appl* 2004; **24**: 48–55.
21. Li X, Gu L, Zhang S, *et al*. A hybrid collision response in a haptic virtual surgery system. Proceedings of the 6th International Special Topic Conference on Information Technology Applications in Biomedicine, Tokyo, Japan, 2007; 1311–1334.
22. Teschner M, Heidelberger B, Muller M, *et al*. A versatile and robust model for geometrically complex deformable solids. Proc Computer Graphics International, 2004. Crete, Greece, 312–319.
23. Baraff D. Fast contact force computation for non-penetrating rigid bodies. Proceedings of the 21st Annual Conference on Computer Graphics and Interactive Techniques, Orlando, Florida, USA, 1994; 23–34.

Appendix I

The domain of response force we worked out is nearly $(-\infty, +\infty)$, and the aim of the mapping step was to map it into $(-7.9, 7.9)$, so the arc tangent function chosen for its domain was $(-\infty, +\infty)$ and the range was $(-\frac{\pi}{2}, \frac{\pi}{2})$, which can be zoomed to $(-7.9, 7.9)$. The mapping was

DOI: 10.1002/rcs

defined as follows:

$$\begin{cases} scaledForceX = \dfrac{2 \cdot \arctan(forceX) \cdot forceThreshold}{\pi} \\ scaledForceY = \dfrac{2 \cdot \arctan(forceY) \cdot forceThreshold}{\pi} \\ scaledForceZ = \dfrac{2 \cdot \arctan(forceZ) \cdot forceThreshold}{\pi} \end{cases} \tag{A1}$$

where *forceX, forceY* and *forceZ* are the response forces in three dimensions, *scaledForceX, scaledForceY* and *scaledForceZ* are the respective corresponding forces after mapping and *forceThreshold* is the biggest force sent to the PHANToM Desktop; clearly, therefore, $forceThreshold \leq 7.9$.

However, there still exists a problem; for example, if the force in the x dimension is 1000 N and in the y dimension 100 N, in fact the force in the x dimension is a dominating force. However, if we use equation (6) directly, we get $scaledForceX = arctan(1\,000) = 1.5698$ N, and $scaledForceY = arctan(100) = 1.5608$ N, which are nearly the same. To solve this problem, we find the biggest force in three dimensions, compute its arc tangent value, then scale the other two forces, so equation (A1) is improved as follows:

$$\begin{cases} scaledForceX = \dfrac{2 \cdot forceX \cdot \arctan}{\dfrac{(forceMax) \cdot forceThreshold}{\pi \cdot forceMax}} \\ scaledForceY = \dfrac{2 \cdot forceY \cdot \arctan}{\dfrac{(forceMax) \cdot forceThreshold}{\pi \cdot forceMax}} \\ scaledForceZ = \dfrac{2 \cdot forceZ \cdot \arctan}{\dfrac{(forceMax) \cdot forceThreshold}{\pi \cdot forceMax}} \end{cases} \tag{A2}$$

where *forceMax* represents the biggest absolute value of the forces in 3 dimensions. Equation A2 fits most of the scenario; nevertheless, in a scenario in which forces are all very strong, e.g. all >100 N, the arc tangent values will all be near to $\frac{\pi}{2}$ and after scaling the results will be near to the *forceThreshold*, which is not the ideal result we want to see. The perfect result is that most of the arc tangent value is around $\frac{\pi}{4}$, which indicates that most of the feedback forces are around *forceThreshold/2*. Here equation A3 is used:

$$\begin{cases} scaledForceX = \dfrac{2 \cdot forceX \cdot \arctan[(forceMax \cdot \pi)/(4 \cdot commonForce)] \cdot forceThreshold}{\pi \cdot forceMax} \\ scaledForceY = \dfrac{2 \cdot forceY \cdot \arctan[(forceMax \cdot \pi)/(4 \cdot commonForce)] \cdot forceThreshold}{\pi \cdot forceMax} \\ scaledForceZ = \dfrac{2 \cdot forceZ \cdot \arctan[(forceMax \cdot \pi)/(4 \cdot commonForce)] \cdot forceThreshold}{\pi \cdot forceMax} \end{cases} \tag{A3}$$

where *commonForce* is the force that appears with a high frequency. When $forceMax = commonForce$:

$$\frac{forceMax \cdot \pi}{4 \cdot commonForce} = \frac{\pi}{4}$$

where *forceMax* is scaled to *forceThreshold/2*.

DOI: 10.1002/rcs