Research on Force Feedback Technology in Virtual Surgery Simulation

Zu Chao[1],Xie Jie-zhen[1], Wang Bo-liang[1], Min Xiao-ping[1]*
1.Department of Computer Science, Xiamen University, Xiamen, 361005, China;
zuchao123@163.com; xjz@xmu.edu.cn; blwang@xmu.edu.cn;pass123456@sohu.com

Abstract

In this paper, the collision detection and response that based on the OBB (Oriented Bounding Boxes) tree structure was realized in virtual surgery simulation system for nasopharyngeal carcinoma. The point of contact location and the vector of contact surface can be acquired. The force effect in virtual surgery simulation was improved. A twice detection method was proposed to improve the phenomenon of the force puncture, and a synchronize mechanism was adopt to improve the stability of the force output. Simulation results show that the improved algorithm has been enhanced in the force puncture and stability of the force, and the force effect is more authentic.

1. Introduction

Virtual surgery simulation enable physicians to interact with flexibly customized simulated environments based on visual, auditory and haptic feedback without potentially harmful contact with real patients. For this reason, virtual surgery simulation has attracted considerable attention as a key technology for the advancement of medical treatments and improvement of quality of human life. In the field of medicine, virtual surgery simulation system are applied for uses such as education, therapy and rehabilitation, procedural training, surgical planning, rehearsal, and interpretative support[1,2].

Haptic feedback is especially important in delicate surgical pressures requiring fine sensations, especially when slightly excessive pressure can injure a patient. The ability to touch the virtual objects in the environments is one factor that provides a more realistic experience and is particularly suited to virtual training applications. A significant constraint on any application exploiting the human haptic system is the update rate that must be obtained. If a rate of 1000 Hz is not achieved the haptic feedback becomes unstable causing undesirable vibrations to be perceived by the user. This ultimately compromises the realism of the simulation. In this paper multi-threaded architecture is presented. Haptic threads and visual threads work separately and share resources. But there are still some problems, such as how communicate between the threads that have such a big difference in frequency.

For at least the last 10 years haptic feedback applications have enabled the user to interact with the virtual environment via a single point. Initially restrictions were imposed which only permitted relatively simple rigid scenes. Zilles et al.[3] and Ruspini et al.[4] produced techniques for single point rendering algorithms incorporating a “God Object” or “Virtual Proxy” to ensure that the contact point displayed is not shown to be penetrating the objects in the scene and to keep a contact history. In single point rendering algorithms the user is able to control the haptic device to interact with objects via a single point. The two methods mentioned above employ a second point in addition to the HIP. This additional point (“Virtual Proxy” or “God Object”) is constrained to the surface of the objects in the virtual environment. The constrained point is located at the same point that the HIP would occupy if the contact between the object in the scene and the HIP were infinitely stiff. When moving in free space the constrained point and the HIP are collocated. These approaches employ different strategies to determine contacts and to keep track of the constrained point. A force can then be returned to the user based on the penetration of the HIP inside the virtual object, whilst the constrained point is graphically rendered for visualization on the surface of the object.

McNeely.[5] developed a method to allow a three-dimensional object to be attached to the haptic interface point. The collision detection was computed by distributing a series of test points over the surface of the tool object. These test points were then tested against a voxel grid encompassing the scene. Later,

978-1-4244-2511-2/08/$25.00 ©2008 IEEE

Gregory[6] extended the "Virtual Proxy" method to a six degree of-freedom force feedback algorithm for the haptic rendering of rigid tools, represented by polygonal models. Their algorithm uses the Lin-Canny closest points algorithm[7] as the basis for the collision detection and uses a predictive approach to minimize the penetrations between the probe and the virtual environment. Johnson[8] have also produced a six degree-of-freedom haptic rendering algorithm for rigid polygonal models. Their method computes local minimum distances using Spatialized Normal Cone Hierarchies[9], and then applies preventative forces to ensure the objects do not penetrate each other.

2. Structure of the virtual surgery simulation system

As previously mentioned, collisions between instruments and organs are especially important to consider in the haptic displays of virtual surgery simulations. Our group addresses this challenge by developing a virtual surgery simulation system for nasopharyngeal carcinoma and using it to evaluate the proposed model.

Figure 1 illustrates the the structure of our virtual surgery simulation system for nasopharyngeal carcinoma. The figure shows the three system components used to simulate object-organ interaction with a haptic display: a data processing unit, graphic display unit, and haptic display unit. The data processing program is coded in Visual C++ and run on a general computer with XEON 3.20GHz CPU and 2GB of main memory. The system is equipped with a Phantom Desktop device produced by SensAble Technologies.

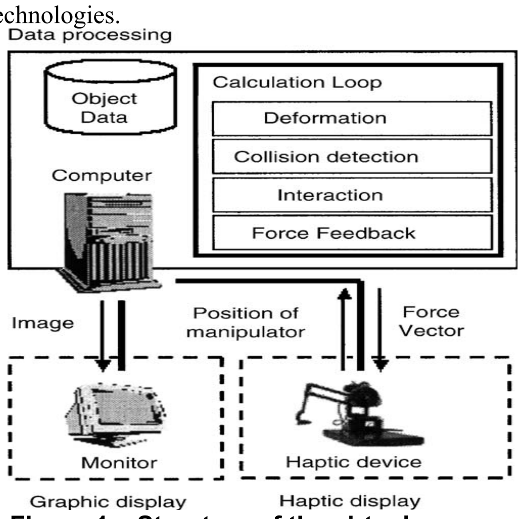

Figure 1. Structure of the virtual surgery simulation system

Organ object data are generally obtained from medical imaging modalities such as CT, MRI, and cross-sectional images. We semi-automatically segment the data and divide them into tetrahedral elements for finite element computations, and store geometry data such as vertices (nodes), tetrahedrons, and surface triangles (polygons). Real-time simulation with haptic display is achieved by applying a static and linear model and fast computation techniques such as condensation[10] and Hirota's method[11]. Inverse stiffness matrices representing the stiffness of objects are pre-computed and stored for the physics simulation. The manipulator position is updated from a haptic device. The reaction force, a parameter calculated in the physics simulation, is conveyed to the user kinesthetically via the haptic device.

3. Calculation of collision between a deformable object and a rigid scene

The calculation of collision consists of the following procedures:
1. Detection of colliding elements.
2. Calculation of temporary displacements.
3. Temporary deformation.
4. Calculation of temporary surface forces.
5. Calculation of actual displacements.

The temporary deformation and the temporary surface forces are based on the finite element method, a method which produces accurate deformation and reaction forces. The detection of the colliding elements depends on the collision detection algorithm.
The steps in each procedure are outlined below.

3.1. Detection of colliding elements. Collisions are checked by testing whether a node has moved inside a polygon of another object. If collisions are detected on both sides of the objects, the following procedures are carried out.

3.2. Calculation of temporary displacements. If a collision between a node X of object A and a polygon S of object B is detected, the polygon S is displaced perpendicularly, as shown in figure 2. The vector of temporary displacement $\overrightarrow{UtempB}$ is:

$$\overrightarrow{UtempB} = \overrightarrow{FX} \qquad (1)$$

where F is the perpendicular foot of nodes X to S. Temporary displacements of nodes P, Q, and R are $\overrightarrow{UtempB}$. The new positions, P', Q', and R' are as follows.

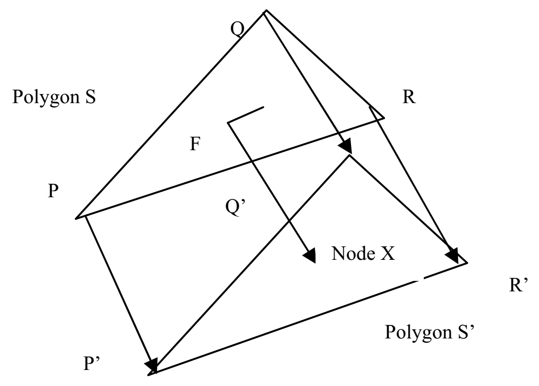

Figure 2. Temporary displacement of the polygon

$$\begin{aligned}\overrightarrow{P'} &= \vec{P} + \overrightarrow{UtempB} \\ \overrightarrow{Q'} &= \vec{Q} + \overrightarrow{UtempB} \qquad (2) \\ \overrightarrow{R'} &= \vec{R} + \overrightarrow{UtempB}\end{aligned}$$

In the same way, a temporary displacement $\overrightarrow{UtempA}$ is calculated and given to the polygon of object A.

3.3. Temporary deformation. The temporary deformation is calculated based on the finite element method using Hirota's model[12] with temporary displacements.

3.4. Calculation of temporary surface forces. A temporary surface force $\vec{f}$ of polygon S is defined as an average vector of nodal forces on P', Q', R':

$$\vec{f} = \frac{\overrightarrow{f_P} + \overrightarrow{f_Q} + \overrightarrow{f_R}}{3} \qquad (3)$$

where $\overrightarrow{f_P}, \overrightarrow{f_Q}, \overrightarrow{f_R}$ are the nodal forces on P', Q', R'. This calculation gives the temporary surface forces based on the stiffness.

3.5. Calculation of actual displacements. Actual displacements of colliding elements of the object A, B are calculated as $|\overrightarrow{f_B}| : |\overrightarrow{f_A}|$ and given to the elements, as surface forces indicate the degree of resistance to invasion of colliding objects.

To visualize the contact boundary of the colliding area realistically, the simulation displaces the polygon of only one object and the node of another object, instead of displacing the polygons of both objects. The sum of the displacements of the polygon and the node are $\overrightarrow{UtempB}$. The node is positioned on the polygon. Accordingly, this method avoids invasion and separation between the polygon and node. It also reduces the computation of the finite element method, as the displacement of only one polygon instead of two results in the displacement of fewer nodes (three nodes are displaced for each polygon). Displacements $\overrightarrow{U_A}, \overrightarrow{U_B}$ of colliding elements of object A, B are as follows:

$$\overrightarrow{U_A} = -\frac{|\overrightarrow{f_B}|}{|\overrightarrow{f_A}| + |\overrightarrow{f_B}|} \overrightarrow{UtempB} \qquad (4)$$

$$\overrightarrow{U_B} = \frac{|\overrightarrow{f_A}|}{|\overrightarrow{f_A}| + |\overrightarrow{f_B}|} \overrightarrow{UtempB} \qquad (5)$$

where $\overrightarrow{UtempB}$ is a vector of temporary displacements, $\overrightarrow{f_A}$ and $\overrightarrow{f_B}$ are surface forces on the colliding polygons of object A, B, respectively, $\overrightarrow{U_A}$ is given to the node of object A, and $\overrightarrow{U_B}$ is given to the component nodes of the polygon of object B.

This method enables us to determine the deformations in the collision area with due consideration of the physical properties of the colliding objects.

4. The Improvement of Force Effects

4.1. The improvement of puncture phenomenon. Since the collision detection is not timely, the collision detection procedures have not detected the collision the moment when the two objects contact, but after some time it have detected and lead to two objects penetrate each other. So this phenomenon called puncture or penetration.

Puncture phenomenon should be strongly avoided in a virtual surgery simulation. However, due to the limitative hardware conditions, the collision detection always be delayed. So it is inescapable for the puncture phenomenon.

In order to solve this problem, besides continuously improve the collision detection algorithms to shorten the collision detection time, people also seek other ways to resolve this problem.

In the virtual surgery simulation system for nasopharyngeal carcinoma, we have designed a method called twice detection. In this virtual simulation, collision detection is between the haptic device's end-effector and the trigonal faces of skull in the scene. When there is no contact between the haptic device's end-effector and the faces, two possible situations may emerge: one is the haptic device's end-effector outside the body of the skull, in this situation there is assuredly no collision; the other is the haptic device's end-effector inside the body of the skull, in this situation puncture phenomenon occurs. At this moment, we should judge if the direction of movement is in accords with the normal of the contacted face, otherwise, get the location of the haptic device's end-effector the moment before as LastPoint, then determine whether there is a collision, if so, it means that this moment the LastPoint is in the body of the skull and puncture occurs. The former moment there must have been a collision that has not been detected, so we should put out the force to stop the haptic device's end-effector going forward. Because the haptic thread is updated at 1000 Hz and the current point position is got about every 1 ms, Therefore, even if puncture phenomenon has happened, the haptic device's end-effector also can be seen touching trigonal faces because the distance with the former location is too short. At this moment a force output can be delayed and do not influence the authenticity of the force output.

Table.1 and Table.2 show the experimental results before and after improvement. Because the experimental results relate to the speed of the haptic device's end-effector, the faster it moves, the shorter time it stays in the faces, the greater possibility the collision detection delays and puncture happens. Therefore, the speed must guarantee invariable, and only under this premise the experimental results is effective. 10 collisions in the experiments are divided into a group, and we should record puncture times that occurred in 10 collisions and the average speed of the haptic device's end-effector, ensuring that the speed changeless. Such experiments conducted a total of 10 groups, the last column shows the total number of puncture of 10 groups and the average speed.

From tables 1 and 2 we can see that the difference between the average speed of 144.0 improved before and the average speed of 148.2 improved is so small that we can affirm the speed before and after improving is identical. So we can see that the puncture improved before took place 14 times in100 collisions, and the puncture improved took place 4 times in 100 collisions. The probability of a puncture occurs reduces 10%. We can get the mathematical expectation E η1 improved before and the mathematical expectation E η2 improved:

$E\eta 1 = 0*P(\eta=0) + 1*P(\eta=1) + 2*P(\eta=2) = 1.4$;

$E\eta 2= 0*P(\eta=0) + 1*P(\eta=1) = 0.4$

Obviously E η2 is less than E η1, so we can announce the effects improved is better than improved before.

4.2. Improvement of the stability of the force. In the virtual surgery simulation, the haptic refresh frame rate is very high (between 300 and 1000Hz), while the visual refresh frame rate is much lower compared to haptic refresh frame rate (20Hz). So, it is very important issue how to coordinate these two threads with such big different refresh frame rates. The delays in communication between threads will lead to the instability of force output and a larger vibration in the contact point. In virtual surgery simulation system for nasopharyngeal carcinoma, spring force model is used to calculate force output. Due to big different refresh frame rates between threads, delay in communication happens, resulting in the unstable force output.

Table 1. Experimental results of puncture number before improvement

Collision times	10	10	10	10	10	10	10	10	10	10	100
Puncture times	1	2	2	2	0	1	2	2	0	2	14
Speed mm/s	131.1	156.8	143.3	142.9	136.5	143.4	150.7	137.0	144.5	151.7	144.0

Table 2. Experimental results of puncture number after improvement

Collision times	10	10	10	10	10	10	10	10	10	10	100
Puncture times	1	1	0	1	0	0	0	0	1	0	4
Speed mm/s	156.8	155.2	141.9	151.9	139.7	140.8	152.8	144.3	151.8	146.8	148.2

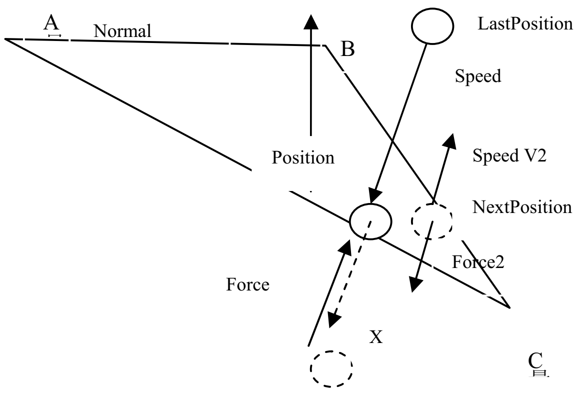

Figure 3. Sketch map of the force feedback

We have adopted a synchronize mechanism which is provided by OpenHpticsTM toolkit Development Kit to solve this problem. We use two separate but simultaneous threads: the visual thread and haptic thread, to realize of graphics rendering and haptic rendering separately. The haptic thread updates the current location of the equipment and the calculates force with 1KHz rates. Visual thread captures the current location with 30Hz rates. These two threads share geometry and topology information of the object through a shared synchronizer, as well as such as the movement information of the equipment.

Collision detection works in the visual thread. The refresh rate in visual thread is 30Hz, therefore, the collision detection executes every 30ms. Because the results of collision detection will be used in the haptic thread, the result of the collision detection must be passed to haptic thread timely. In order to pass the results to haptic thread (1000Hz), we can use the function ***hdScheduleSynchronous***() in synchronize mechanism to transport messages between different threads.

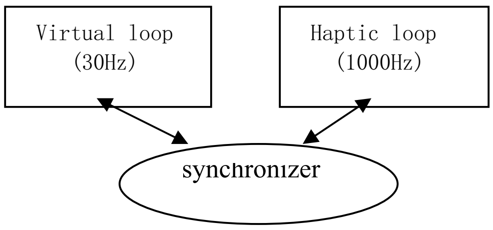

Figure 4. Thread relations of the system

```
In visual loop (30 Hz):
{
    Get the current location of equipment;
    Convert the position into the world coordinate system (Position-> pt);
    Collision detection   ioo = InsideOrOutside ( pt );
    Pass ioo to haptic thread by HDCALLBACK function(PassMassageCallback);
    hdScheduleSynchronous(PassMassageCallback, &ioo, HD_DEFAULT_SCHEDULER_PRIORITY);
}
```

```
In haptic loop (1000 Hz):
{
    If ( ioo = -1 )       // Collision is detected
          bRenderForce = TRUE;
    Else if (dot of Speed and Normal <0)
      // Collision is not detected but puncture occurs
    { If ( ioo2 = -1)
        // Puncture occurs, force should be output
       bRenderForce = TRUE;
    }
Else
     bRenderForce = FALSE;
    Calculate the vector X between position and anchor;
    If (dot of X and Normal < 0)
          bRenderForce = FALSE;
    if (bRenderForce = TRUE;)
          Calculate the force and output;
}
```

Ioo2 is the result of collision detection about last Position. From figure 3, there is a unnecessary force (Force2) when the position of the device leave the contacted surface due to the delay of collision detection. Therefore, we should compare judge the dot of the vector x and the normal with 0, if it is less than 0, Force2 is set to 0 to avoid unnecessary force.

From figure 6 we can see that the pre-improved force feedback curve is unstable and fluctuate strongly. of a greater force of the sudden change of power output, and chart the value of negative forces on the edge direction Change, The improved force feedback curve becomes more smooth and stable.

The method is also used in the virtual surgery simulation system for liver, which is shown in the figure 7 and 8, making the force output more stable.

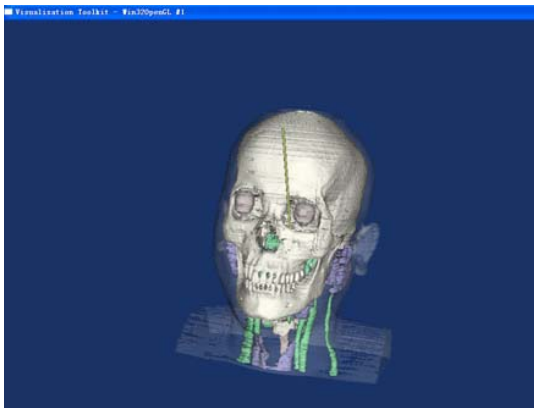

Figure 5. Virtual surgery simulation system for nasopharyngeal carcinoma

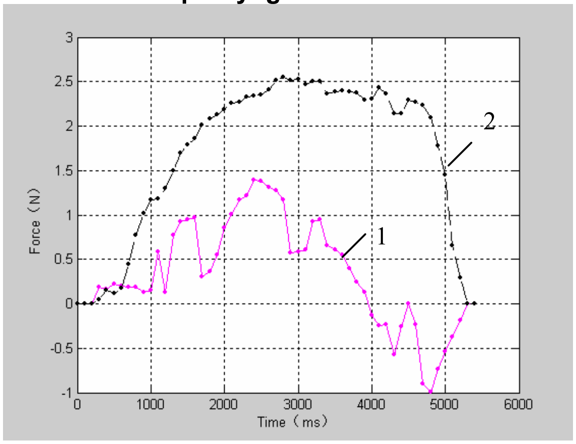

Figure 6. The force feedback curves before and after the improvement in the Virtual surgery simulation system for nasopharyngeal carcinoma (Curve 1 represent result before improvement, curve 2 represent result after improvement)

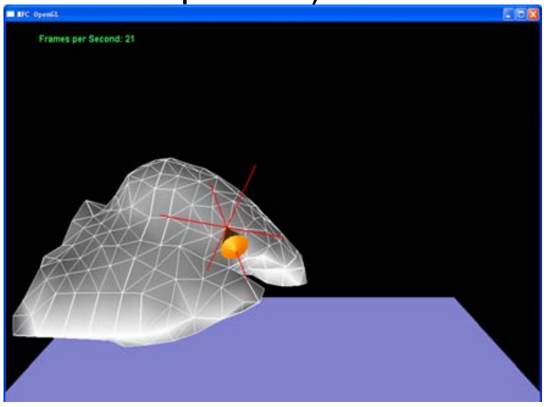

Figure 7. Virtual surgery simulation for liver cancer

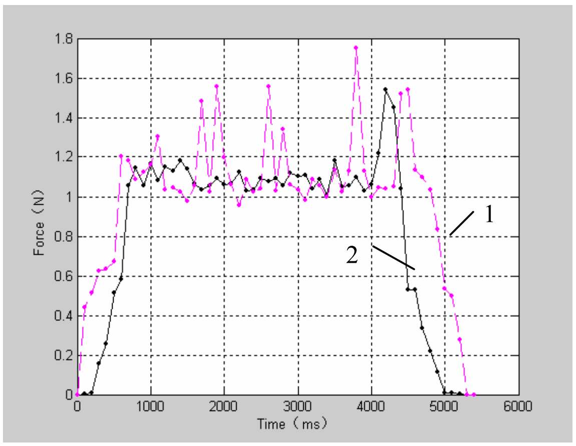

Figure 8. The force feedback curves before and after the improvement in the Virtual surgery simulation system for liver cancer

The simulation environment is:

XEON CPU3.20GHz, 2G memory, NVIDIA Quadro FX4400 graphics accelerator card ;

Sensable's PHANTOM Desktop force feedback device ;

VTK(Visualization Toolkit) ;

OpenHptics^{TM} toolkit.

5. Conclusions

In virtual surgery simulation system for nasopharyngeal carcinoma, the collision detection and response that based on the OBB (oriented Bounding boxes) tree structure was realized .And the point of contact location and the vector of contact surface can be returned. Due to a large amount of data, collision detection will be delayed which can cause a puncture phenomenon and unstable of force output. A twice detection method was proposed to improve the phenomenon of the force puncture, and a synchronize mechanism was adopt to improve the stability of the force output. Experimental results show that the improved algorithm has been enhanced in the force puncture and stability of the force so that the force effect is more authentic.

Supported by the National Nature Science Foundation of China (60601025,60701022 and 30770561)

References

[1] R. Satava, in: J.D. Westwood, et al. (Eds.), Medical Virtual Reality: The Current Status of the Future, in Medicine Meets Virtual Reality, IOS Press, Amsterdam, 1996, pp. 100—106.

[2] G. Burdea, P. Coiffet, Virtual Reality Technology, Wiley Interscience, 2003.

[3] Zilles CB, Salisbury JK. A constraint based God－Object method for haptic display. Proceedings of the IEEE conference on intelligent robots and systems, 1995, pp. 146－51.

[4] Ruspini DC, Kolarov K, Khatib O. The Haptic Display of Complex Graphical Environments. Computer graphics proceedings, Siggraph. 1997, Los Angeles.

[5] McNeely W, Puterbaugh K, Troy J. Six degree-of-freedom haptic rendering using voxel sampling. Proceedings of the ACM SIGGRAPH, 1999, pp. 401－8.

[6] Gregory A, Mascarenhas A, Ehmann S, Lin MC, Manocha D. Six degree-of-freedom haptic display of polygonal models. IEEE Visualization, 2001.

[7] Lin MC, Canny JF. A fast algorithm for incremental distance computation. In: Proceedings of the IEEE international conference on robotics and automation, April 1991, pp. 1008－14.

[8] Johnson DE, Willemsen P. Six degree-of-freedom haptic rendering of complex polygonal models, 11th symposium on haptic interfaces for virtual environments and teleoperated systems, 2003.3.

[9] Johnson DE, Cohen E. Spatialized Normal Cone hierarchies, In: Proceedings of the 2001 ACM symposium on interactive 3D graphics, Research Triangle Park. March 19－21, 2001. pp. 129 - 34.

[10] M. Bro-Nielsen, Finite element modeling in surgery simulation, J. IEEE 86 (3) , 1998 . pp.490 - 503.

[11] K. Hirota, T. Kaneko, Haptic representation of elastic objects, presence: teleoperators and virtual environments, MIT Press 10 (5) , 2001 . pp .525 - 536.

[12] K. Hirota, T. Kaneko, Haptic representation of elastic objects, presence: teleoperators and virtual environments, MIT Press 10 (5) , 2001. pp .525 - 536.