Comparison between FEM and BEM for Real-time Simulation

Y. M. Tang[1], A. F. Zhou[2] and K. C. Hui[3]

[1]The Chinese University of Hong Kong, ymtang@acae.cuhk.edu.hk
[2]The Chinese University of Hong Kong, afzhou@acae.cuhk.edu.hk
[3]The Chinese University of Hong Kong, kchui@acae.cuhk.edu.hk

ABSTRACT

Computer simulation of object deformation has wide applications in areas such as movies, computer graphics, computer games, etc. Numerous methods have been proposed to simulate deformable objects. A common practice of simulating deformable objects is to use physically-based approaches which include the mass-spring system and the finite element method (FEM). The mass-spring system only gives a coarse estimation of object deformation whereas the FEM requires generating solid volumetric elements which is a tedious and time consuming process. The use of boundary element method (BEM) allows objects to be deformed without generating solid volumetric elements.

In order to achieve real-time deformation, all these methods require time-consuming pre-computation process. In this paper, a comparison is made between the FEM and the BEM techniques. We propose to adopt linear elements of boundary element technique for real-time applications. The method not only allows physically-based and real-time deformation, it also requires much shorter time for the pre-computation process. Experimental results show that the time required for the pre-computation process and real-time deformation can be enhanced significantly by adopting linear elements of BEM.

Keywords: boundary element method, physically-based modeling, linear elements, stiffness matrix

1. INTRODUCTION

Techniques for simulating deformable objects have been widely studied in the context of computer graphics and used in movies, computer games and other activities. Over the past two decades, various approaches for object simulation have been introduced. However, it remains a great challenge in computer graphics because of the conflicting demands of accuracy and real-time deformation. One example is surgery simulation [1], [2] which requires both accurate deformation and real-time interactions. However, human organs and tissues have very complex form, modeling real-time deformation is only possible using very simplistic models. Approaches for modeling object deformation can be divided into two categories: physically based and non-physically based. Non-physically based approaches such as free-form deformation (FFD) employ purely geometric techniques for model deformation. Those techniques rely on the skill of the users rather than the physical properties of an object. In this paper, focus is put on real-time deformation using physically based approach.

A good survey of various techniques in object deformation can be found in [3]. Physically-based modeling methods allow explicit deformation of objects according to their physical properties, which have been widely used for realistic simulation of deformable objects. Among the physically-based modeling methods, the most popular one is the mass-spring system [4-6] because of its simplicity and capability to achieve real-time performance. The mass-spring system approximates an object model by modeling the domain as a set of mass points connected by springs. Despite the simplicity in the formulation of the mass-spring system, the method is not suitable for various applications like surgery simulation, gait physical therapy, etc. because a large number of nodes is required for accurate simulation.

Nowadays, a substantial amount of work has been devoted to the use of FEM [7-10] which provides a more accurate physical simulation. However, the major limitations of this method are the complexity in implementing FEM and the

high computational cost of evaluating deformation. Besides, FEM requires generating solid volumetric elements which is a tedious and time consuming process.

The BEM has very long history in engineering analysis applications [11], [12]. However, it is not widely used in computer graphics for simulating deformable objects [13], [14]. Actually, it takes advantages of both accurate deformation and computational efficiency. Instead of solving stresses and deformations of a volumetric model, BEM determines displacements and tractions of a boundary model. Thus, it leads to much smaller number of unknowns required to be solved while accurate deformation at the boundary of the model can be obtained. By adopting BEM, the same boundary mesh can be used for deforming and rendering an object. The boundary mesh can also be constructed easily with popular graphics packages. This eliminate the efforts on generating solid volumetric elements of FEM. James *et al.* [14] has demonstrated that deformable objects can be simulated in real-time and attained physical accuracy. For simplification, their calculations are based on the constant element case of BEM. However, a major drawback of using constant element is that accurate deformation results can only be achieved when the model is fine enough and with smooth surface. A fine model implies large mesh size of the model, this increases the computation time of the pre-computation process and the real-time simulation significantly. To reduce the computation complexity, decreasing size of the stiffness matrix is a direct approach.

This paper compares FEM and BEM by considering the time required for the pre-computation process and the real-time deformation process. Experimental results demonstrated that a fast and accurate real-time simulation of deformable objects can be achieved by adopting linear elements of boundary element technique because it requires the stiffness matrix in much smaller size than FEM and constant elements of BEM. The technique allows not only physically-based and real-time deformation, but also requires much shorter time for the pre-computation process.

In this paper, we:

- discuss the advantages of employing linear elements of BEM for object simulation in computer graphics applications;
- and compare among FEM, constant elements of BEM and linear elements of BEM according to speed of the pre-computation process and deformation for real-time simulation.

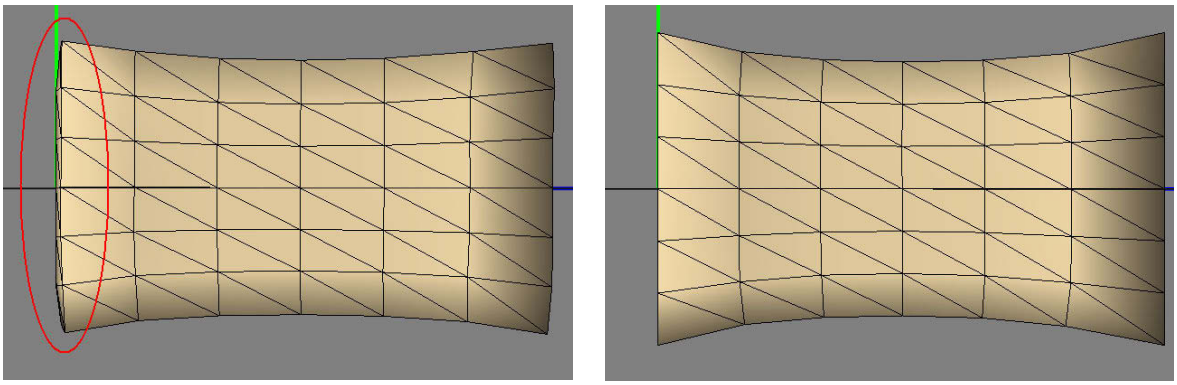
The remainder of this paper is organized as follows. Section 2 compares different boundary elements. The boundary element technique is discussed in section 3. Section 4 discusses the pre-computation process and the algorithm to achieve real-time deformation in this paper. Experimental results are given in section 5, and section 6 concludes the paper.

2. COMPARSION OF BOUNDARY ELEMENTS

Consider a 3-D model with boundary Γ, BEM discretize the boundary into a finite number of segments which are called boundary elements. The boundary elements are part of its external surface and are usually of two types: quadrilateral and triangular. The simplest and the most common type for deformable objects in computer graphics are triangular elements. In this paper, we assume the boundary of the objects is divided into triangular elements. These can be constant, linear, quadratic, or higher order. The points where displacements and tractions are considered are called nodes. Displacements and tractions, and nodes on the boundary are called boundary values and nodal values respectively. In the case of constant elements, the boundary is approximated by a triangular mesh. The node of an element is located at the centre of the triangle and the boundary value is assumed to be constant for this element. In the case of linear elements, the boundary is approximated by a triangular mesh and the nodes are located at the vertices of the triangles. The boundary values vary linearly between the nodal values. For the case of quadratic elements, each element has three nodes for each edge and the variation of the boundary values is quadratic.

Among different types of element, quadratic or higher order elements allow more accurate interpolation of the boundary values over each element. Nevertheless, additional information is required to obtain the boundary values between the vertices which are always difficult to acquire in computer graphics. These extra boundary values increases the size of the stiffness matrix which increases the time for the pre-computation process and is undesired for real-time deformation. Since our research mainly focus on speeding up the pre-computation process and real-time deformation, quadratic or higher order elements are not adopted.

Considering the case of constant elements, nodes are located at the centre of each element. Deformations at each vertex are usually computed by averaging the nodal values that share the vertex. The calculation seems to be reasonable if the mesh is regular, however, problems exist for irregular mesh. Suppose the left face of a cube is fixed and a deformation is applied on the right face, undesired deformation occurs at the edge of the cube using constant elements (see Fig. 1(a)). Fig. 1(b) shows that the use of linear elements gives correct deformation result because nodes of linear elements are located at the vertices of an object.



(a) (b)

Fig. 1. Deformation of a cube with fixed left face using: (a) constant elements; (b) linear elements

Constant element is also known as discontinuous element. Fig. 2 shows the approximation of a function by discontinuous constant elements. It is evident that discretization of the boundary using constant elements requires larger number of segments to maintain an accurate approximation. In spite of the fact that the deformation process can be speed up by using constant elements, deformation is not accurate as discussed above. Linear elements make a good balance between accuracy and real-time deformation. Linear elements are also known as continuous elements. Fig. 3 shows the approximation of a function by continuous linear elements. Comparing with Fig. 2, it can be easily seen that linear elements can approximate the exact function more accurately. For a 3-D boundary element mesh, the number of linear elements which is equal to the number of vertices of the model, is smaller than the number of constant elements which is the same as the number of triangular faces of the model. As a result, linear elements can achieve both higher accuracy and faster deformation comparing with constant elements. In the sections that follow, we will discuss the linear element case for modeling and deformation.

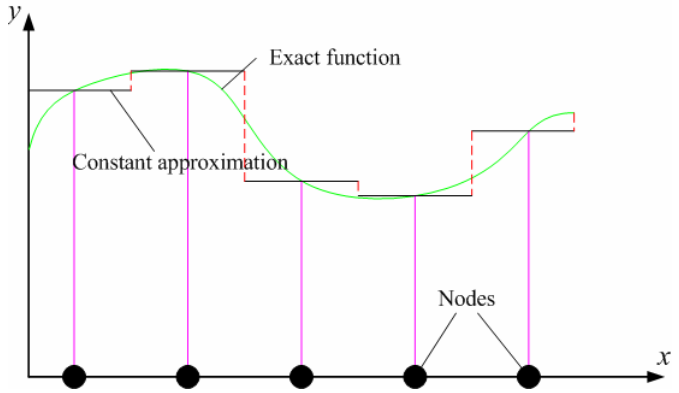


Fig. 2. Approximation of the exact function by discontinuous constant elements

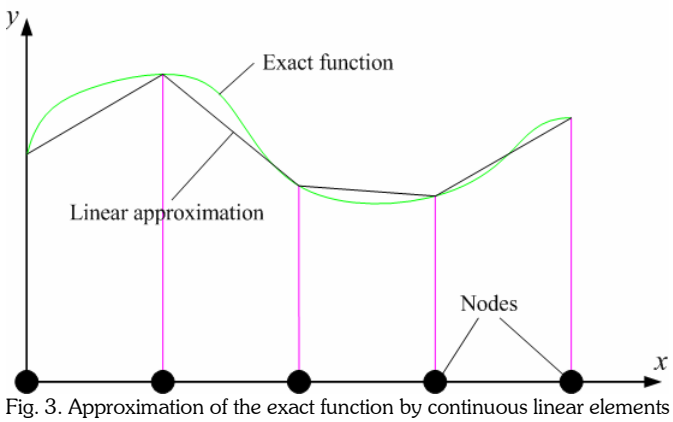


Fig. 3. Approximation of the exact function by continuous linear elements

3. THE BOUNDARY ELEMENT TECHNIQUE

For the boundary integral equation at the k^{th} source point and omitting the body forces, the equation in matrix form can be simplified as

$$c_k u_k + \int_\Gamma t_k^* u d\Gamma = \int_\Gamma u_k^* t d\Gamma \,, \tag{1}$$

where the u_k^* and t_k^* represent the fundamental solutions of displacements and tractions. The smoothness coefficient, displacements and tractions applied at the k^{th} source point are denoted by c_k, $u_k = \left(u_{kx}, u_{ky}, u_{kz}\right)^t$ and $t_k = \left(t_{kx}, t_{ky}, t_{kz}\right)^t$ respectively. The smoothness coefficient depends on the smoothness properties of the boundary at the source point [15]. In most cases, the smoothness coefficient does not require to be solved explicitly. It can be solved indirectly such as using rigid body consideration.

Note that the smoothness coefficient and the fundamental solutions are known in Eqn. (1). The physical properties of the models using BEM can be easily tuned by the user as it requires only two well-known material properties, the Possion's ratio, υ and the modulus of elasticity, E which can be easily found in many material handbooks.

The idea of BEM is to employ a numerical approach to discretize the boundary of the model into a series of elements. In the m^{th} element, displacements u_m and tractions t_m are approximated from the n^{th} nodal displacements u_m^n and tractions t_m^n using the shape function Φ. The discretization of Eqn. (1) can be rewritten by converting the surface integrals into sums of integrals over each boundary element, which yields

$$c_k u_k + \sum_{m=1}^{M}\sum_{n=1}^{N}\left(\int_{\Gamma_m} t_k^* \Phi^n d\Gamma\right) u_m^n = \sum_{m=1}^{M}\sum_{n=1}^{N}\left(\int_{\Gamma_m} u_k^* \Phi^n d\Gamma\right) t_m^n \,, \tag{2}$$

where N is the number of nodal points in each element which is equal to one for constant elements and three for linear elements, and Γ_m is the surface of the m^{th} element.

Grouping all the integrals with same node together, Eqn. (2) yields

$$\sum_{l=1}^{L} h_{kl} u_l = \sum_{l=1}^{L} g_{kl} t_l \,, \tag{3}$$

where L is total number of nodal elements.

Assembling all the elements together and using matrix notation, Eqn. (3) can be written as

$$\mathbf{HU} = \mathbf{GT} \,, \tag{4}$$

where $\mathbf{U} = \left[u_{1x} \;\; u_{1y} \;\; u_{1z} \;\; u_{2x} \;\; u_{2y} \;\; u_{2z} \;\; \cdots \;\; u_{Lx} \;\; u_{Ly} \;\; u_{Lz}\right]^t$, $\mathbf{T} = \left[t_{1x} \;\; t_{1y} \;\; t_{1z} \;\; t_{2x} \;\; t_{2y} \;\; t_{2z} \;\; \cdots \;\; t_{Lx} \;\; t_{Ly} \;\; t_{Lz}\right]^t$,

$\mathbf{H}$ is a function of h_{kl} and

$\mathbf{G}$ is a function of g_{kl}.

Evaluation of integrals of the fundamental solutions is the most crucial aspect in BEM because it exhibit singularities at certain points in the elements. Depending on the relative position between the source and the field points, the integrals can be classified into singular and non-singular types.

- Non-singular integrals
  For the non-singular case, the source point and the field point lies in different node. The non-singular integrals may be evaluated analytically, however the process is very complicated and computationally impractical. Thus, numerical integration approaches like Gaussian quadrature [16-18] is commonly used which gives satisfactory result.

- Singular integrals
  In this case, the source lies on the element where the integration is performed. Define *r* as the distance between the source point and the field points. There exists two types of singularities, the kernel g_{kl} contains functions of order 1/*r* is weakly singular and the kernel h_{kl} contains functions of order $1/r^2$ is strongly singular which only exist as *Cauchy* principal values [19]. Depending on the type of singularity, different techniques are used to evaluate the integrals. Numerical quadrature formulas for integration over triangles and squares with 1/*r* singularity presented by Pina *et al.* [20] and Cristescu and Loubignac [21] are simple and computationally efficient. For the evaluation of the strongly singular terms h_{kk}, an indirect approach that considers rigid body movements of an object is commonly used in BEM.

4. PRE-COMPUTATION AND REAL-TIME DEFORMATION

After evaluation of the integrals, unknowns in Eqn. (4) are displacements and tractions of the nodes. Although in linear element case, the nodal displacements are continuous, a discontinuity in the traction still occurs at that node if the boundary of the model is not smooth. However object deformation in computer graphics requires only displacement to be determined, technique that used in FEM to determine object deformations is still applicable in BEM. Eqn. (4) can be expressed as

$$\mathbf{KU} = \mathbf{T}, \tag{5}$$

where **K** is the stiffness matrix.

To achieve real-time simulation, pre-computation is an essential process in computer graphics. The pre-computation process is to determine the stiffness matrix, **K** and its inverse, $\mathbf{K}^{-1}$. Size of the stiffness matrix and its inverse are equal to the number of nodes of the model. They are usually very large in size and the pre-computation process is extremely time-consuming. To decrease time required for the pre-computation process, the only way is to decrease the number of nodes of the model. After **K** and $\mathbf{K}^{-1}$ have been pre-computed, those terms in the pre-computed matrices can be rearranged and used to determine the unknown displacements.

Denoting $\mathbf{U}_K$ as displacements of the elements where deformation is applied, $\mathbf{T}_K$ as tractions at the free elements, $\mathbf{U}_U$ and $\mathbf{T}_U$ as displacements and tractions of the elements that are going to be determined. Rearranging the terms, **K** can be divided into four sub-matrices and Eqn. (5) becomes

$$\begin{bmatrix} \mathbf{K}_{00} & \mathbf{K}_{01} \\ \mathbf{K}_{10} & \mathbf{K}_{11} \end{bmatrix} \begin{bmatrix} \mathbf{U}_U \\ \mathbf{U}_K \end{bmatrix} = \begin{bmatrix} \mathbf{T}_K \\ \mathbf{T}_U \end{bmatrix}. \tag{6}$$

Since tractions at all free elements are zero, displacements of unknown elements can be determined by

$$\mathbf{U}_U = -\mathbf{K}_{00}^{-1}\mathbf{K}_{01}\mathbf{U}_K. \tag{7}$$

Given the boundary constraints and displacements at certain elements of the model, displacements of unknown elements can be computed by Eqn. (7). By determining the unknown displacements, objects can be deformed accordingly.

To perform deformation at interactive frame rates, Eqn. (7) has to be updated within one-tenth of a second. The bottleneck of computation is on updating the inverse of sub-matrices of stiffness matrix which are usually very large in size. In the case of specified boundary conditions, displacements and tractions of unknown elements are fixed. Storing the computed matrix $\left[-\mathbf{K}_{00}^{-1}\mathbf{K}_{01}\right]$ allows real-time deformation to be possible as it includes simply a matrix vector multiplication process. However, boundary conditions usually change and create other contact points with the model. The dimension of $\mathbf{U}_U$ and $\mathbf{T}_U$ will vary depending upon the number of contact points which prescribe $\mathbf{U}_K$ and $\mathbf{T}_K$. This requires reevaluating $\mathbf{K}_{00}^{-1}$.

If the number of known displacements is larger than the number of unknown displacements, $\mathbf{K}_{00}^{-1}$ is computed using Eqn. (7). Otherwise, special technique is used to achieve fast computation by updating $\mathbf{K}_{00}^{-1}$ based on the sub-matrices of $\mathbf{K}^{-1}$. Assuming $\mathbf{K}$ and $\mathbf{K}^{-1}$ are pre-computed, $\mathbf{K}\mathbf{K}^{-1}=\mathbf{I}$ can be expressed in terms of their sub-matrices, such that

$$\begin{bmatrix}\mathbf{K}_{00} & \mathbf{K}_{01}\\ \mathbf{K}_{10} & \mathbf{K}_{11}\end{bmatrix}\begin{bmatrix}\mathbf{D}_{00} & \mathbf{D}_{01}\\ \mathbf{D}_{10} & \mathbf{D}_{11}\end{bmatrix}=\begin{bmatrix}\mathbf{I} & 0\\ 0 & \mathbf{I}\end{bmatrix}. \tag{8}$$

Eliminating $\mathbf{K}_{01}$, we have

$$\mathbf{K}_{00}^{-1}=\mathbf{D}_{00}-\mathbf{D}_{01}\mathbf{D}_{11}^{-1}\mathbf{D}_{10}. \tag{9}$$

As seen from Eqn. (9), $\mathbf{K}_{00}^{-1}$ can be computed from the sub-matrices of the pre-computed $\mathbf{K}^{-1}$. The size of matrix $\mathbf{D}_{11}$ is determined by the number of known displacements whereas the size of matrix $\mathbf{K}_{00}$ is determined by the number of unknown displacements. If the number of known displacements is less than the number of unknown displacements, the size of $\mathbf{D}_{11}$ is smaller than $\mathbf{K}_{00}$. Since the time for computing matrix inverse dominates the computation time of the deformation process, speed of evaluating deformation is enhanced by computing $\mathbf{D}_{11}^{-1}$.

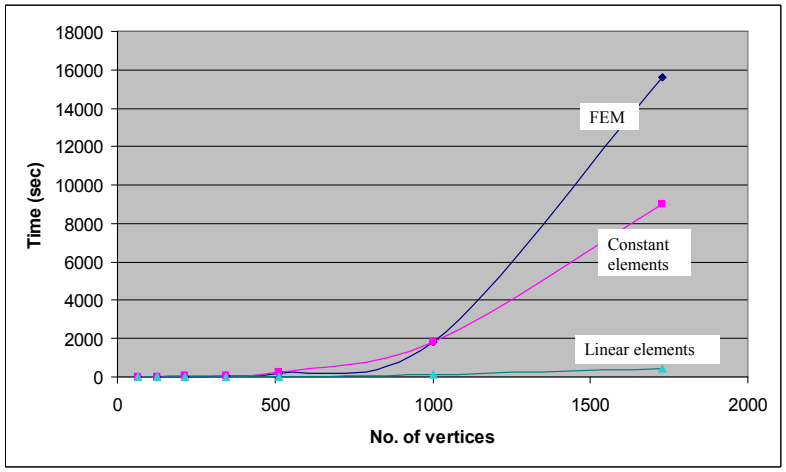
Considering a model with *N* elements in which there are *n* elements with known displacements. Since the time complexity for evaluating the inverse of a $M\times M$ matrix is $O(M^3)$, the time complexity for evaluating $\mathbf{K}_{00}^{-1}$ and $\mathbf{D}_{00}^{-1}$ are thus $O((N-n)^3)$ and $O(n^3)$ respectively. Therefore, if $N >> n$, Eqn. (9) is preferred.

5. EXPERIMENTAL RESULTS

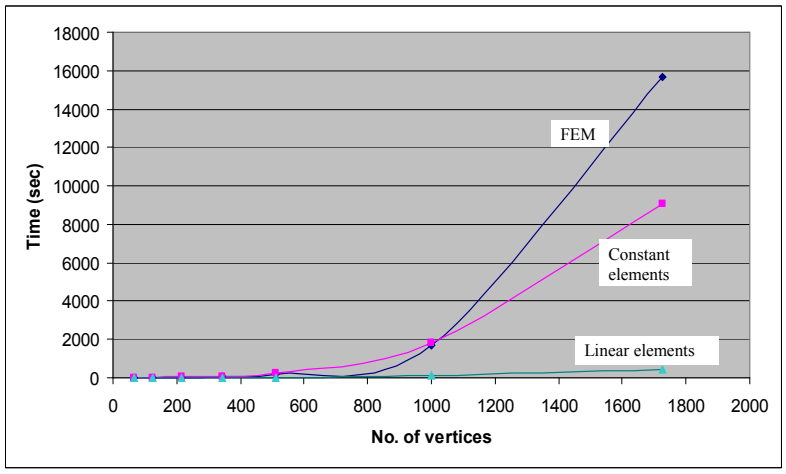
In this section, we compare performances of FEM and BEM using a cube model. This experiment was divided into three parts. In the first part, the time required for the pre-computation process was compared among FEM, constant elements of BEM and linear elements of BEM. The performance of real-time deformation using Eqn. (9) to update $\mathbf{K}_{00}^{-1}$ among FEM, constant elements of BEM and linear elements of BEM was compared in the second part. In the third part, the performance of real-time deformation with and without using Eqn. (9) to update $\mathbf{K}_{00}^{-1}$ was compared. Linear elements of BEM were applied to deform the model. In order to study the relationship between different number of known displacements and the performance of the method, different percentages of known displacement were assigned for each part of the experiment. In our experiment, a computer with a Pentium4 3.2GHz CPU and 2G Bytes memory was used.

5.1 Comparing Time Required for the Pre-computation Process

This experiment compares time required for the pre-computation process using FEM, constant elements of BEM and linear elements of BEM. The pre-computation process includes computing the stiffness matrix and its inverse. The setup of this experiment is described as follows. A set of cubes with different number of vertices were first generated. Then, two cases were considered. For the first case, displacements of 10% nodes were known and a certain value of displacement was assigned. Displacements of 30% nodes were assigned to a certain value for the second case. In each case, FEM, constant elements and linear elements were applied for pre-computation.



(a)



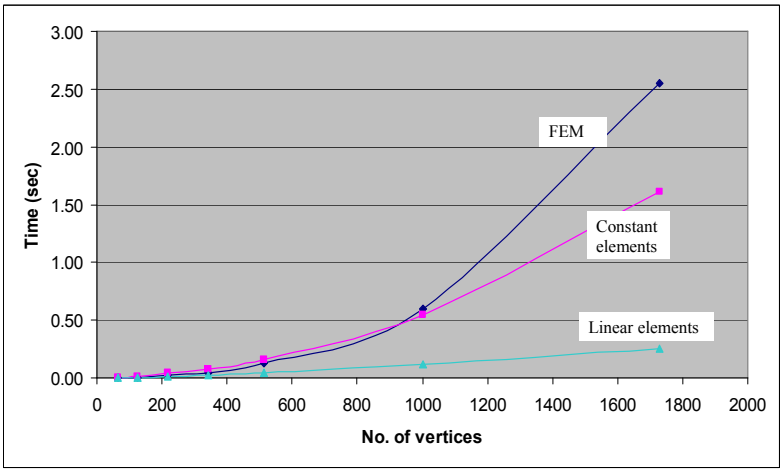
(b)

Fig. 4. Comparing time required for the pre-computation process using Eqn. (9) with known displacements of: (a) 10%; (b) 30%

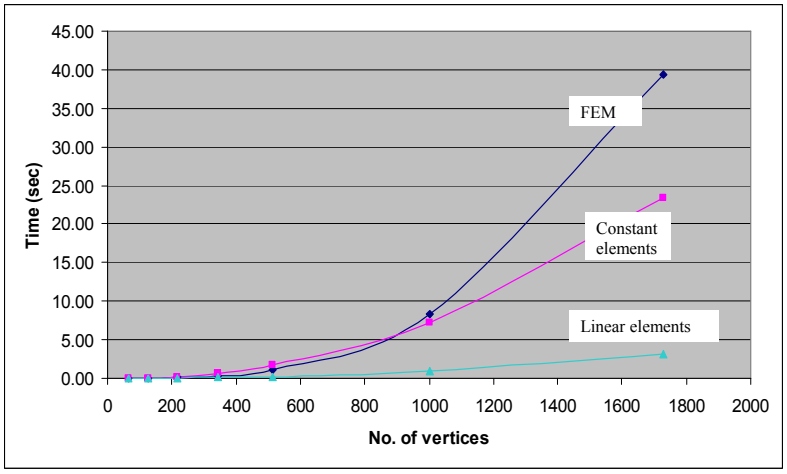
Fig. 4 compares the time required for the pre-computation process using FEM, constant elements of BEM and linear elements of BEM. As seen from Fig. 4., the time required for the pre-computation process using FEM is a little bit shorter than constant elements for objects with small number of vertices. As the number of vertices increases, the pre-computation time required using FEM increases rapidly. Much longer time is required for constant elements for cubes with more than 1000 vertices. However, linear elements demonstrated a very good performance on pre-computing small and large number of vertices of the cube. On the other hand, linear elements take less than 200s for pre-computing the model with 1000 vertices whereas FEM and constant elements spent around 30 minutes! Time required for the pre-computation process is approximately the same for 10% and 30% of known displacements. This agrees that time required for the pre-computation process is independent with the number of known displacements.

5.2 Comparing Time Required for Real-time Deformation

In this part, performance of interactive simulation using FEM, constant elements of BEM and linear element of BEM were compared. Suppose a set of cubes with different number of vertices were first generated. Then, two cases were introduced. For the first case, displacements of 10% nodes were known and a certain value of displacement was assigned. Displacements of 30% of nodes were assigned to a certain value for the second case. In each case, FEM, constant elements and linear elements were applied to deform the cube and Eqn. (9) was used to update $\mathbf{K}_{00}^{-1}$ during deformation.



(a)



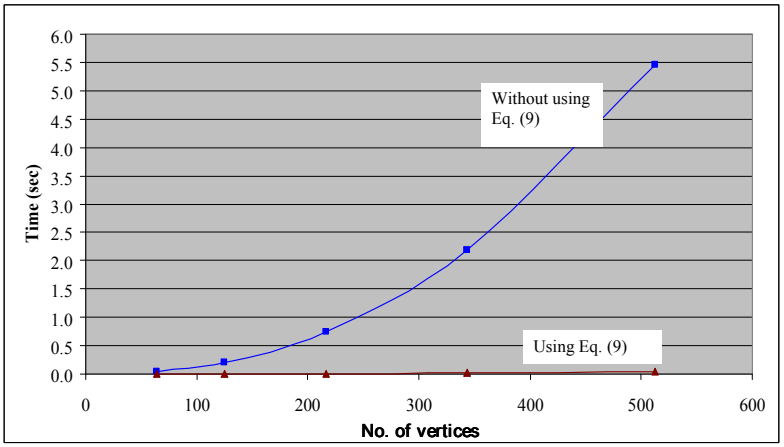
(b)

Fig. 5. Comparing time required for real-time deformation using Eqn. (9) with known displacements of: (a) 10%; (b) 30%

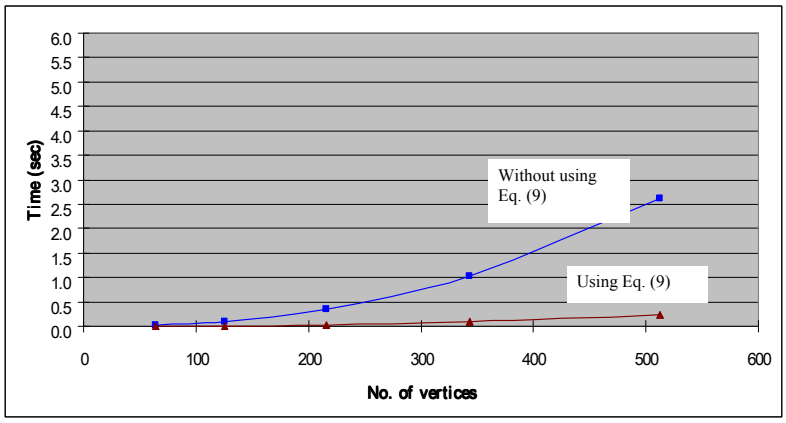
From Fig. 5, performance of real-time deformation is similar to the simulation results as in the first part of the experiment. The time required for interactive simulation using FEM is less than constant elements for a cube with vertex number less than about 950. As the number of vertices further increases, the time required for real-time simulation for constant elements becomes less than the FEM. Besides, linear elements demonstrated a very good performance on real-time simulation for small and large number of vertices of the cube. For a cube with 1000 vertices, the time required for real-time simulation using linear elements is only 0.1s which is around one-fifth of the time required for FEM and constant elements! However, as the number of known displacements increases, the time required for real-time simulation for FEM and BEM increases significantly.

5.3 Comparing Performance With and Without Using Eqn. (9)

To test the performance of real-time deformation, Fig. 6 compares the time required for computing $\mathbf{U}_U$ with and without using Eqn. (9) to update $\mathbf{K}_{00}^{-1}$ using linear elements of BEM. A set of cubes with different number of vertices were first generated. In order to study the relationship between different number of known displacements and the computational time, 10% and 30% of total number of nodes were randomly selected, and their displacements were assigned to a certain value. Linear elements were applied for real-time simulation.



(a)



(b)

Fig. 6. Performance with and without using Eqn. (9) for real-time deformation using linear elements with known displacements of: (a) 10%; (b) 30%

Fig. 6 shows the comparison of performance for real-time deformation under 10% and 30% of nodes with known displacements respectively. As shown in Fig. 6, the computational time required to compute $\mathbf{U}_U$ using Eqn. (9) is very small. Real-time performance can thus be achieved. However, as the percentage of nodes with known displacements increases, the computational time using Eqn. (9) increases, whereas the computational time decreases without using Eqn. (9).

6. CONCLUSION

In this paper, a comparison between FEM, constant elements of BEM and linear elements of BEM for real-time simulation has been demonstrated. Several experiments that compare speed of the pre-computation process and real-time deformation have been performed. Employing linear elements, speed of the pre-computation process and deformation for real-time simulation can be enhanced significantly. Moreover, an accurate deformation can be achieved using BEM because deformations are based on the physical properties of the object. Besides, boundary model for BEM analysis can be easily constructed with popular graphics packages. In order to allow interactive simulation, emphasis is put on speeding up the matrix inversion process. Experimental results showed that speed of the matrix inversion process are enhanced significantly using Eqn. (9).

7. ACKNOWLEDGEMENT

This work is partially supported by a grant from the Research Grants Council of the Hong Kong Special Administration Region (Project No. CUHK4197/04E) and an Innovation and Technology Fund from the

Innovation and Technology Commission of the Hong Kong Special Administration Region (Project No. ITS/091/02).

8. REFERENCES

[1] Bro-Nielsen, M., and Cotin, S., Real-time volumetric deformable models for surgery simulation using finite elements and condensation, *Computer Graphics Forum*, Vol. 15, 1996, pp 57-66.

[2] Cotin S., Delingette H. and Ayache N., Real-time elastic deformations of soft tissues for surgery simulation, *IEEE Trans. on Visualization and Computer Graphics*, Vol. 5, No. 1, 1999, pp 62-73.

[3] Gibson S. F. and Mirtich B., *A survey of deformable models in computer graphics*, Technical Report TR-97-19, Mitsubishi Electric Research Laboratories, Cambridge, MA, November 1997.

[4] Desbrun M., Schröder P. and Barr A., Interactive Animation of Structured Deformable Objects, *Proceedings of Graphics Interface*, 1999, pp 1-8.

[5] Luciani A., Jimenez S., Florens J.-L., Cadoz C. and Raoult O., Computational Physics: a Modeler-Simulator for Animated Physical Objects, *Eurographics*, Elsevier, Vienne (Autriche), 1991.

[6] Miller G., The Motion Dynamics of Snakes and Worms, *Computer Graphics*, Vol. 22, No. 4, 1988, pp 169-178.

[7] Hui K. C. and Wong. N. N., Hands on a virtually elastic object, *The Visual Computer*, Vol. 18, No. 3, 2002, pp 150-163.

[8] Gourret J.-P., Thalmann N. M. and Thalmann D., Simulation of object and human skin deformations in a grasping task, *Computer Graphics*, Vol. 23, No. 3, 1989, pp 21-29.

[9] Kenneth H. H., Earl A. T. and Ted G. B., *The finite element method for engineers*, New York: Wiley, 1995.

[10] Zienkiewicz O. C. and Taylor R. L., *The finite element method*, New York: McGraw-Hill, 1991.

[11] Katsikadelis J. T., *Boundary elements: theory and applications*, New York: Elsevier, 2002.

[12] Brebbia C. A., Telles J. C. F. and Wrobel L. C., *Boundary element techniques: theory and applications in engineering*, Springer-Verlag, 1984.

[13] Hui K. C. and Leung H. C., Virtual Sculpting and Deformable Volume Modelling, *Proceedings of Information Visualisation*, 2002, pp 664-669.

[14] James D. L. and Pai D. K., ArtDefo: Accurate Real Time Deformable Objects, *Proceedings of Computer Graphics*, 1999, pp 65-72.

[15] Beer G. and Watson J. O., *Introduction to finite and boundary element methods for engineers*, New York: Wiley, 1992.

[16] Stroud A. H. and Secrest D., *Gaussian quadrature formulas*, Prentice-Hall, 1966.

[17] Dunavant D. A., High degree efficient symmetrical Gaussian quadrature rules for the triangle, *Int. Journal for Numerical Methods in Engineering*, Vol. 21, 1985, pp 1129-1148.

[18] Cowper G. R, Gaussian quadrature formulas for triangles, *Int. Journal for Numerical Methods in Engineering*, Vol. 7, 1998, pp 405-410.

[19] Beer G., *Programming the boundary element method: an introduction for engineers*, New York: Wiley, 2001.

[20] Pina H. L. G., Fernandes J. L. M. and Brebbia C. A., Some numerical integration formulae over triangles and squares with a 1/R singularity, *Applied Mathematical Modeling*, Vol. 5, 1981, pp 209-211.

[21] Cristescu M. and Loubignac G., Gaussian quadrature formulas for functions with singularities in 1/r over triangles and quadrangles in *Recent Advances in Boundary Element Methods* (C.A. Brebbia, Ed.), Pentech Press, London, 1978, pp 375-390.