An Accelerated BEM Approach for the Simulation of Deformable Objects

A. F. Zhou[1], K. C. Hui[2], Y. M. Tang[3] and Charlie C. L. Wang[4]

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

ABSTRACT

This paper introduces an approach for speeding up the simulation on deformable objects with the boundary element method (BEM). According to the relationship between a matrix and its inverse, the inverse operation on an objective matrix can be completed by calculating the inverse of another matrix with smaller dimension. An update procedure on inverse calculation for three cases is analyzed to avoid directly computing the matrix inverse every time when the boundary condition changes. Besides, a fast matrix-matrix multiplication (MMM) method is conducted to further improve the computational speed.

Keywords: boundary element method, matrix inverse update, Strassen-Winograd algorithm

1. INTRODUCTION

Simulation on deformable bodies gains wide applications in these years. Various deformation approaches have been developed in literature [6], among which physically-based modeling technique is a very important one as it is based on realistic physical properties. As far as engineering precision is concerned, the finite element method (FEM) and the boundary element method (BEM) are two representatives. FEM is a very popular method in computational engineering [8, 13]. However, as it requires generating solid meshes from the volume data, the computation complexity is rather high. Comparatively, the boundary element method is more advantageous in this aspect.

When simulating deformable objects, boundary element method is preferred because of its two superiorities: high accuracy and efficiency [16]. In the simulation of low-latency interactive simulation on linear elastic models, the boundary element method is a good choice since the stiffness matrix can be pre-computed. Also, BEM is well suited to be applied in tracking deformable objects by template matching algorithm [9], since it only requires the surface tessellation of a flexible object. For the research of virtual sculpting and volume modeling, boundary element method also exhibits its advantages as a good physically-based deformation technique [12]. Because of these remarkable virtues, we adopt BEM in our research to simulate static linear elastic deformation, which gives an accurate description on small deformations of objects.

In our work, the unknown deformation is obtained based on the external displacement. Compared with the external force, the input displacement can be precisely measured with little difficulty, which makes the implementation much easier. During the simulation, in order to achieve real-time deformation, the key problem is the speed of calculating matrix inverse. A capacitance matrix algorithm (CMA) was proposed in [14]. As matrix dimension is the most influential factor which affects the computational time in our research, some transformation technique will be carried out to decrease the matrix dimensions in this paper. By constructing the relationship between a matrix and its inverse, the inverse operation on an objective matrix could be replaced by calculating the inverse of a matrix with smaller size. After that, the computation is further speeded up by applying a matrix inverse update procedure, with which only small rank perturbation is applied on the original matrix. Three cases of the changes on matrix dimension are considered. In this way, a series of algebraic operations on matrices replace the re-computation of inverse matrix. In addition, as calculating matrix products is also a time-consuming process when it is frequently carried out, Strassen-Winograd (SW) [11] algorithm is incorporated to speed up the matrix-matrix multiplication process. Through these three acceleration techniques, our approach shows significant effects on the improvement of computing speed

Our paper is organized as follows. Section 2 gives a description on the deformation technique by the boundary element method. The local updating technique for computing inversed matrices is discussed in section 3. Section 4 introduces a fast matrix multiplication method. Experimental results and analysis are finally given in section 5.

2. THE DEFORMATION TECHNIQUE BY BOUNDARY ELEMENT METHOD

The equilibrium condition of linear elastic objects with isotropic and homogeneous material properties can be given by the well-known Navier equation [1]. To solve the Navier equation, the finite element method and the boundary element method are two popular approaches. For the later approach, as the displacement and traction can be evaluated directly on the surface mesh of objects, it avoids generating solid mesh from the volume data and improves the computation efficiency.

For an object with boundary Γ, the Navier equation is transformed into a boundary integral equation, which can be expressed as (assume there is no body force)

$$c_{pq}u_{ref} + \int_{\Gamma} u_q t_{pq}^{*} d\Gamma = \int_{\Gamma} t_q u_{pq}^{*} d\Gamma \tag{1}$$

where u_{ref} is the displacement of a reference point on Γ, t_{pq}^{*} and u_{pq}^{*} are the fundamental solutions to the Navier equation. c_{pq} is the smoothness coefficient of the reference point. $t_p = \begin{bmatrix} t_{px} & t_{py} & t_{pz} \end{bmatrix}^T$ and $u_p = \begin{bmatrix} u_{px} & u_{py} & u_{pz} \end{bmatrix}^T$ denote the tractions and displacements of a point on the surface Γ. By approximating Γ with a set of node points, and applying each of the node points as reference point in Eqn.(1), a series of linear equations can be obtained. Expressing the linear equations in matrix form gives

$$\mathbf{Hu} = \mathbf{Gt} \tag{2}$$

where $\mathbf{H}$ is a function of t_{pq}^{*} and c_{pq}, $\mathbf{G}$ is a function of u_{pq}^{*}, $\mathbf{u}$ and $\mathbf{t}$ denote the displacement and traction vectors at the node points of the object respectively. Further detailed information about the boundary element technique can be referred in [1].

We know that, at the known external displacement nodes, the tractions are unknown. On the contrary, the displacement is unknown of a node where the traction is zero. The elements on an object can hence be classified into two groups: vertices with known displacement and vertices with unknown displacement. Partitioning the tractions and displacements according to these two attributes, and dividing $\mathbf{K}$ into four sub-matrices accordingly, we may transform Eqn. (2) into

$$\begin{bmatrix} \mathbf{t}^k \\ \mathbf{t}^u \end{bmatrix} = \mathbf{K} \begin{bmatrix} \mathbf{u}^u \\ \mathbf{u}^k \end{bmatrix} = \begin{bmatrix} \mathbf{K}_{11} & \mathbf{K}_{12} \\ \mathbf{K}_{21} & \mathbf{K}_{22} \end{bmatrix} \begin{bmatrix} \mathbf{u}^u \\ \mathbf{u}^k \end{bmatrix} \tag{3}$$

where $\mathbf{K} = \mathbf{G}^{-1}\mathbf{H}$ is the stiffness matrix, which involves the effects of Poisson's ratio and shear modulus , $\mathbf{t}^k$ and $\mathbf{t}^u$ are the known and unknown force vectors, $\mathbf{u}^u$ and $\mathbf{u}^k$ are the corresponding unknown and known displacement vectors. Define n_k as the number of nodes with known displacements, n_u as the number of nodes with unknown displacements, then $\mathbf{u}^u$ is a vector with n_u components, $\mathbf{u}^k$ is with n_k components. Since the nodes with unknown displacement are those with zero traction, $\mathbf{K}_{11}$ is a square matrix and the dimension is $n_u \times n_u$.

During the deformation, $\mathbf{u}^u$ is calculated based on a set of $\mathbf{u}^k$ and a set of zero traction vectors $\mathbf{t}^k$ at the force nodes, Eqn. (3) gives

$$\mathbf{u}^u = -{\mathbf{K}_{11}}^{-1}\mathbf{K}_{12}\mathbf{u}^k \tag{4}$$

In order to achieve a fast simulation, $\mathbf{K}$ is calculated in the pre-computation process. During every step of the simulation, the boundary constraints may change, which means the displacement of every element may alter between known and unknown. The dimension and elements of matrices $\mathbf{K}_{11}$ and $\mathbf{K}_{12}$ may also be changed. $\mathbf{K}_{11}$ and $\mathbf{K}_{12}$ are thus formed by selecting the corresponding values in matrix $\mathbf{K}$ according to different boundary conditions. ${\mathbf{K}_{11}}^{-1}$ is then computed from $\mathbf{K}_{11}$. If the number of unknown displacements is much larger than that of the known

displacements (i.e., $n_u >> n_k$), the time spent on directly calculating $\mathbf{K}_{11}^{-1}$ will be very long. Aimed to solve this problem, a transformation is carried out.

As we know, $\mathbf{K}\mathbf{K}^{-1} = \mathbf{I}$ can be expressed in terms of their sub-matrices form, which is

$$\begin{bmatrix} \mathbf{K}_{11} & \mathbf{K}_{12} \\ \mathbf{K}_{21} & \mathbf{K}_{22} \end{bmatrix} \begin{bmatrix} \mathbf{D}_{11} & \mathbf{D}_{12} \\ \mathbf{D}_{21} & \mathbf{D}_{22} \end{bmatrix} = \begin{bmatrix} \mathbf{I} & \mathbf{0} \\ \mathbf{0} & \mathbf{I} \end{bmatrix} \tag{5}$$

Since $\mathbf{K}^{-1}$ has already been obtained in pre-computation, $\mathbf{D}_{11}$, $\mathbf{D}_{12}$, $\mathbf{D}_{21}$ and $\mathbf{D}_{22}$ are determined in the same way of $\mathbf{K}_{11}$ and $\mathbf{K}_{12}$. Eliminating $\mathbf{K}_{12}$ in Eqn.(5), we can get

$$\mathbf{K}_{11}^{-1} = \mathbf{D}_{11} - \mathbf{D}_{12}\mathbf{D}_{22}^{-1}\mathbf{D}_{21} \tag{6}$$

where the size of $\mathbf{D}_{22}$ is $n_k \times n_k$. Thus, the unknown displacement can also be calculated by

$$\mathbf{u}^u = (\mathbf{D}_{12}\mathbf{D}_{22}^{-1}\mathbf{D}_{21} - \mathbf{D}_{11})\mathbf{K}_{12}\mathbf{u}^k \tag{7}$$

In most cases, n_u is much bigger than u_k. In this way, we decrease the dimension of the matrix which needs to be inverted. For an $m \times m$ matrix, the time complexity for evaluating its inverse is $O(m^3)$. Then, the time complexity for evaluating $\mathbf{K}_{11}^{-1}$ and $\mathbf{D}_{22}^{-1}$ are respectively $O(n_u^3)$ and $O(n_k^3)$. Therefore, when $n_u >> n_k$, using Eqn.(7) to compute $\mathbf{u}^u$ greatly reduces the computing time.

3. UPDATING THE INVERSE OF MATRICES

From the expression about the unknown displacement, we may find out that it is unavoidable to calculate the inverse of matrices. As this operation is a very time-consuming process, many efforts have been devoted to enhance the computational efficiency [15]. Instead of re-calculating the inverse matrix by Gauss-Jordan method, we prefer to adopt some local update strategy to determine it from previous one. A very common way is introduced in [10]. By using some numerical techniques, it is possible to update the inverse of a matrix after a small rank perturbation based on the original inverse matrix. According to the changes on the matrix dimension during the simulation, three cases may be shown, that is, matrix with invariable, increased and decreased dimensions. Different processing steps are taken in different cases.

3.1 Matrix with Invariable Dimension

During the deformation, there may be changes in the boundary conditions, but the total number of known displacements remains unchanged. Under this circumstance, the values of some rows and columns in matrix $\mathbf{D}_{22}$ are altered, but the dimension remains unchanged. A fast procedure is applied to update the inverse of $\mathbf{D}_{22}$ with invariable dimension.

Assume both $\mathbf{M}_0$ (original matrix) and $\mathbf{M}$ (updated matrix) are invertible $n \times n$ matrices. Suppose only s columns of $\mathbf{M}_0$ change, let $\mathbf{P}_{n\times s}$ be the matrix consists of the replaced columns, and $\mathbf{Q}_{n\times s}$ be the replacement, defining $\delta\mathbf{M} = \mathbf{Q} - \mathbf{P}$, we have

$$\mathbf{M} = \mathbf{M}_0 + \delta\mathbf{M}\mathbf{E}^T \tag{8}$$

where $\mathbf{E}$ is the $n \times s$ sub-matrix of a $n \times n$ identity matrix [14].

Following the form of Woodbury formula [10], if both $\mathbf{A}$ and $\mathbf{I} - \mathbf{V}\mathbf{A}^{-1}\mathbf{U}$ are invertible matrices, there is

$$[\mathbf{A} - \mathbf{U}\mathbf{V}]^{-1} = \mathbf{A}^{-1} + \mathbf{A}^{-1}\mathbf{U}(\mathbf{I} - \mathbf{V}\mathbf{A}^{-1}\mathbf{U})^{-1}\mathbf{V}\mathbf{A}^{-1} \tag{9}$$

Then applying the Woodbury formula on Eqn.(8), we can get

$$\mathbf{M}^{-1} = (\mathbf{M}_0 + \delta\mathbf{M}\mathbf{E})^{-1} = \mathbf{M}_0^{-1} - \mathbf{M}_0^{-1}\delta\mathbf{M}\left[\mathbf{I} + \mathbf{E}^T(\mathbf{M}_0^{-1}\delta\mathbf{M})\right]^{-1}\mathbf{E}^T\mathbf{M}_0^{-1} \tag{10}$$

where $\mathbf{I} + \mathbf{E}^T(\mathbf{M}_0^{-1}\delta\mathbf{M})$ is the matrix that needs to be directly inverted by Gauss-Jordan method and its dimension is $s \times s$ ($s << n$). As the numerical values of $\mathbf{M}_0^{-1}$ have been recorded, and $\delta\mathbf{M}$ and $\mathbf{E}^T$ can be easily obtained from

the change of boundary conditions, it is not difficult to calculate $\mathbf{I}+\mathbf{E}^T(\mathbf{M}_0{}^{-1}\delta\mathbf{M})$. $\mathbf{M}^{-1}$ can then be updated by Eqn.(10).

3.2 Matrix with Increased Dimension

When objects deform, it is quite often that the number of boundary constraints changes. For this case, there are two possibilities: constraints enlarge and constraints shrink. When the first situation occurs, the number of known displacement increases, thus bringing about a rise in the dimension of matrix $\mathbf{D}_{22}$. Therefore, a study on how to update the inverse of a matrix with an increased size is necessary.

When $\mathbf{M}_0 \rightarrow \mathbf{M}$, the size of matrix enlarges from $n \times n$ to $(n+s)\times(n+s)$. In this case, a Schur complement is introduced to complete the inverse update process. Assume that

$$\mathbf{M}_{(n+s)\times(n+s)} = \begin{bmatrix} (\mathbf{M}_0)_{n\times n} & \mathbf{U} \\ \mathbf{V} & \mathbf{D} \end{bmatrix}, \tag{11}$$

according to Duncan [5], $\mathbf{M}^{-1}$ can be expressed as

$$\mathbf{M}^{-1} = \begin{bmatrix} \mathbf{M}_0{}^{-1} + \mathbf{M}_0{}^{-1}\mathbf{U}\mathbf{C}^{-1}\mathbf{V}\mathbf{M}_0{}^{-1} & -\mathbf{M}_0{}^{-1}\mathbf{U}\mathbf{C}^{-1} \\ -\mathbf{C}^{-1}\mathbf{V}\mathbf{M}_0{}^{-1} & \mathbf{C}^{-1} \end{bmatrix} \tag{12}$$

where $\mathbf{C} = \mathbf{D} - \mathbf{V}\mathbf{M}_0{}^{-1}\mathbf{U}$ is an $s \times s$ matrix called Schur complement.

3.3 Matrix with Decreased Dimension

Just as what we have mentioned in the previous section, there is another case that the boundary constraints may shrink, which means the number of known displacement decreases. Thus, the third situation must be considered where the dimension of $\mathbf{D}_{22}$ decreases.

When $\mathbf{M}_0 \rightarrow \mathbf{M}$, the size of matrix reduces from $n \times n$ to $(n-s)\times(n-s)$. Let

$$(\mathbf{M}_0)_{n\times n}{}^{-1} = \begin{bmatrix} \mathbf{M}'_{(n-s)\times(n-s)} & \mathbf{U}' \\ \mathbf{V}' & \mathbf{D}' \end{bmatrix} \tag{13}$$

On the other hand, as

$$\mathbf{M}_0 = \begin{bmatrix} \mathbf{M}_{(n-s)\times(n-s)} & \mathbf{U} \\ \mathbf{V} & \mathbf{D} \end{bmatrix} \tag{14}$$

It is not difficult to have that

$$\begin{bmatrix} \mathbf{M} & \mathbf{U} \\ \mathbf{V} & \mathbf{D} \end{bmatrix}^{-1} = \begin{bmatrix} \mathbf{M}^{-1} + \mathbf{M}^{-1}\mathbf{U}\mathbf{C}^{-1}\mathbf{V}\mathbf{M}^{-1} & -\mathbf{M}^{-1}\mathbf{U}\mathbf{C}^{-1} \\ -\mathbf{C}^{-1}\mathbf{V}\mathbf{M}^{-1} & \mathbf{C}^{-1} \end{bmatrix} \tag{15}$$

Comparing the corresponding items in Eqn. (13) and (15), there is

$$\begin{cases} -\mathbf{M}^{-1}\mathbf{U}\mathbf{C} = \mathbf{U}' \\ -\mathbf{C}^{-1}\mathbf{V}\mathbf{M}^{-1} = \mathbf{V}' \end{cases} \Rightarrow \mathbf{M}^{-1}\mathbf{U}\mathbf{C}\mathbf{V}\mathbf{M}^{-1} = \mathbf{U}'\mathbf{D}'^{-1}\mathbf{V}' \tag{16}$$

Again, comparing the upper-left quarter in Eqn. (13) and (15), we have

$$\mathbf{M}' = \mathbf{M}^{-1} + \mathbf{M}^{-1}\mathbf{U}\mathbf{C}^{-1}\mathbf{V}\mathbf{M}^{-1}, \tag{17}$$

which can be easily deduced into

$$\mathbf{M}^{-1} = \mathbf{M}' - \mathbf{U}'\mathbf{D}'^{-1}\mathbf{V}' \tag{18}$$

where $\mathbf{D}'$ is an $s \times s$ matrix.

For all the above three conditions, $\mathbf{M}^{-1}$ can be calculated form $\mathbf{M}_0{}^{-1}$ and some perturbation matrices based on $\mathbf{M}_0$. The size of the matrix needs to be inverted by traditional Gauss-Jordon way is reduced to s x s. Thus, the update process can be completed quickly.

4. THE IMPLEMENTATION OF FAST MATRIX MULTIPLICATION ALGORITHM

In the previous section, we improve the efficiency of calculating the matrix inverse by applying some algebraic operations on matrices, namely, addition, subtraction and multiplication, where the matrix-matrix multiplication is a frequently involved operation. Normally, this is completed by one of the Basic Linear Algorithm Subroutines (BLAS) [2]. When huge matrices are multiplied together, the time spending on calculating the products in the traditional way will be extremely long. Aimed at this problem, some algorithms are designed to obtain a high performance Matrix-Matrix Multiplication (MMM) [4, 17]. In our research, GEMMW [3] algorithm is adopted.

In general, the direct calculation of the products of two $n \times n$ matrices

$$(\mathbf{AB})_{ij} = \sum_{k=1}^{n} a_{ik} b_{kj} \tag{19}$$

requires $O(n^3)$ operations [7]. In contrast, Strassen-Winograd's algorithm only costs $O(n^{\log_2 7})$ operations, thus is more asymptotically efficient. Strassen-Winograd's algorithm is realized in the following way. Define $\mathbf{A}$ and $\mathbf{B}$ as matrices of dimensions $m \times k$ and $k \times n$ respectively (m , n and k are even numbers), $\mathbf{A}$, $\mathbf{B}$ and $\mathbf{C} = \mathbf{AB}$ can be divided into four equally sized blocks

$$\begin{bmatrix} \mathbf{C}_{11} & \mathbf{C}_{12} \\ \mathbf{C}_{21} & \mathbf{C}_{22} \end{bmatrix} = \begin{bmatrix} \mathbf{A}_{11} & \mathbf{A}_{12} \\ \mathbf{A}_{21} & \mathbf{A}_{22} \end{bmatrix} \begin{bmatrix} \mathbf{B}_{11} & \mathbf{B}_{12} \\ \mathbf{B}_{21} & \mathbf{B}_{22} \end{bmatrix} \tag{20}$$

To calculate matrix $\mathbf{C}$, instead of 8 multiplications and 4 additions on the sub-matrices of $\mathbf{A}$ and $\mathbf{B}$ in general method, Strassen-Winograd's algorithm recursively forms the products of two matrices in 7 matrix multiplications and 15 additions. Detail of the algorithm can be found in [11].

Based on this idea, Douglas developed a portable level 3 BLAS Winograd variant of Strassen's matrix multiply algorithm, which is called GEMMW. Unlike other methods which also adopt SW algorithm, GEMMW can be applied on non-square matrices. And it does not restrict the matrix dimensions to 2^k (k is a natural number). Besides, GEMMW implements Strassen-Winograd with a minimum of extra storage.

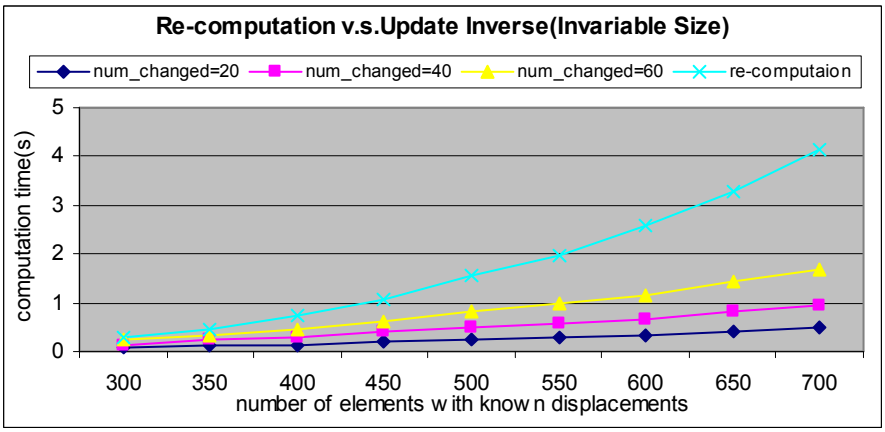
However, as we adopt C style data structure (matrix data are stored in row order) in the simulation, while GEMMW is designed in Fortran style (matrix data are stored in column order), data transformation is required to modify the standard GEMMW to be applicable in our application.

5. EXPERIMENTAL RESULTS

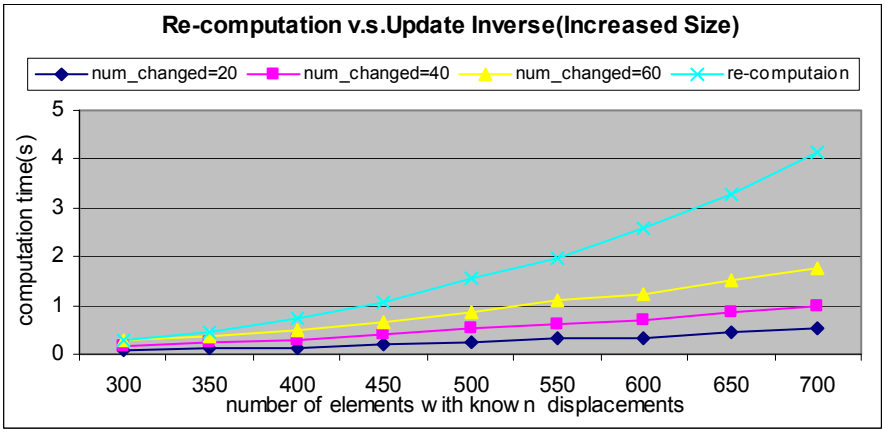
As stated in section 3, the matrix inverse is the most time-consuming step. Hence, the performance of the simulation system is directly affected by the matrix size. Eqn.(7) can effectively reduce the dimension of matrix which needs to be inverted, thus greatly shaving off the time cost on this process. The advantage of Eqn.(7) has been clearly elucidated in [16] by experiments and analysis. In this section, we will put the emphasis on verifying the efficiency of the local update procedure for inversed matrices and Strassen-Winograd algorithm in our applications. The experiments consist of three parts. In the first part, the performance with and without applying the local inversing procedure is tested. The difference between the computation time with and without applying SW algorithm is presented in the second part. Finally, an example of the deformation on a footwear model by the boundary element method is demonstrated. All the tests are performed on a PC with Pentium4 3.0 GHz CPU and 1G Bytes memory.

5.1 Comparing Performance with and without Applying Update Matrix Inverse Procedure

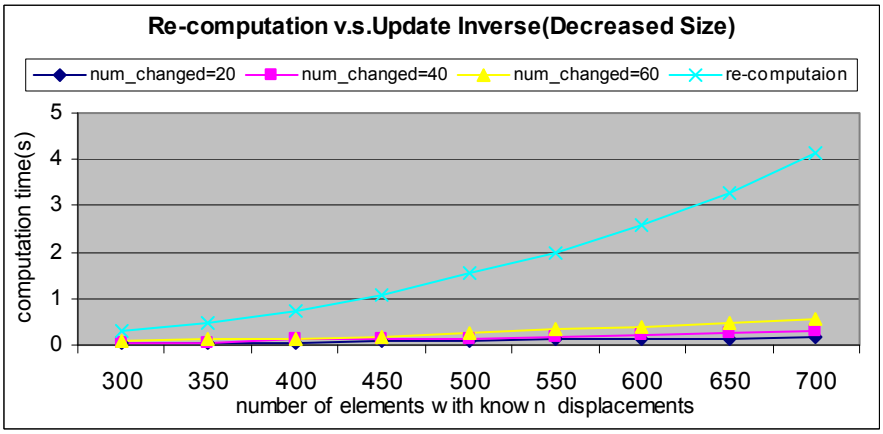
As we know, the simulation time is affected by the total amount of meshes on a geometric model to a large extent. Even for the same object with the same mesh size, different boundary constraints, namely, different numbers of known and unknown displacement elements, will cause different performance. Besides, for the update on the inverse matrix under the three situations discussed in section 3, as the dimension of the matrix which needs to be inverted directly by Gauss-Jordan method is $s \times s$, the number of changed boundary constraints also has certain influence on the ultimate simulation efficiency. Based on these factors, experiments are conducted to measure the performance of a surface model whose total number of elements is 2100, with the amount of elements with known displacement varying form 300 to 700. For a given number of elements with known displacements, the computation time on matrix inverse by the traditional Gauss-Jordan method is compared with our accelerated approach. In order to find out how the number of changed boundary constraints affects the simulation performance, different cases are also tested when s =20, 40 and 60.



(a) invariable dimension



(b) increased dimension



(c) decreased dimension

Fig. 1. Computation time with/without matrix inverse update method.

Fig. 1(a). shows the computation time when there is no dimensional change, but only numerical values changes with the matrix. From the figure, it can be found out that when the update process on matrix inverse is applied, the speed is greatly improved comparing to the conventional method (i.e., re-computing). What's more, we can also see that the less is the number of changed boundary constraints, the faster is the update process completed. This conclusion is also

applicable for other two cases, just as what can be concluded from Fig. 1(b). and Fig. 1(c)., where the matrix dimension is increased and decreased respectively.
Comparing the above three figures, we may also conclude that under the equivalent simulation condition, when the matrix size is reduced, the time required to implement the update procedure is much smaller than that under the other two situations.

5.2 Comparing Performance with and without Applying SW Algorithm
Fig. 2. shows the effect when SW algorithm is used. This experiment is completed on a cube model with 1000 elements. When the percentage of the elements with known displacement varies form 10% to 40%, which means the number of unknown displacements increases from 600 to 900, there is a decreasing trend in the computation time of both general matrix-matrix multiplication and Strassen-Winagrad algorithm. With the aid of Strassen-Winagrad algorithm, about half of the time required for the conventional method is saved. From the same figure, we may also find out that the slope of the red curve is not as steep as that of the blue one. When the number of unknown varies form 600 to 900, the computation time by the general MMM method changes from more than 5 seconds from less than 2 seconds, while the time by the SW algorithm alters from about 2 seconds to nearly 1 second. This means that the efficiency of SW algorithm is not so sensitive to the variety of the matrix size, and thus it is more stable. However, when the dimension of matrix is smaller than certain value [3], the SW algorithm may lose its superiority to general MMM method. Just as what can be seen from the following figure, when the percentage of the number of known displacement elements to the total element number becomes lower, there is a tendency that the two curves will converge.

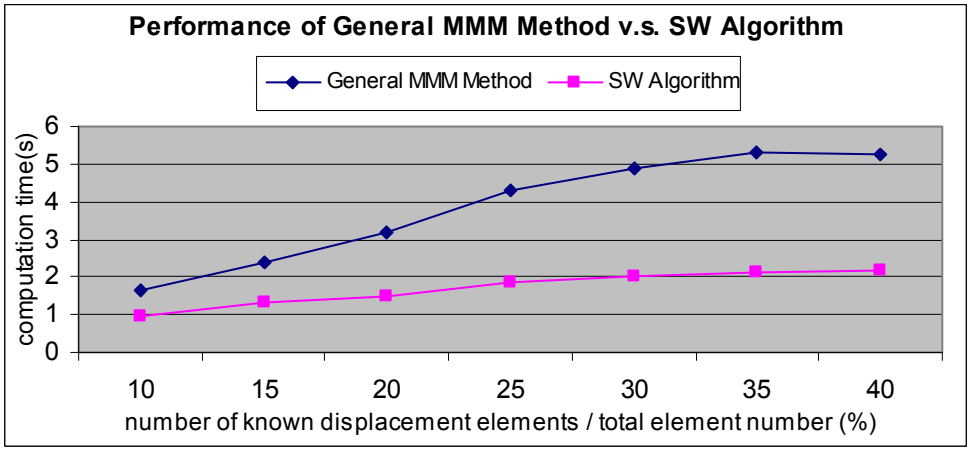
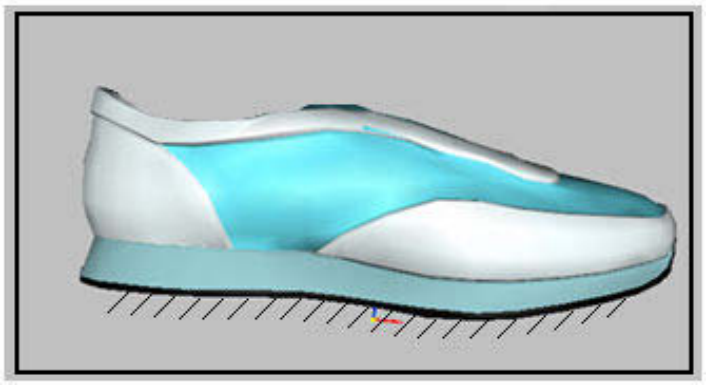


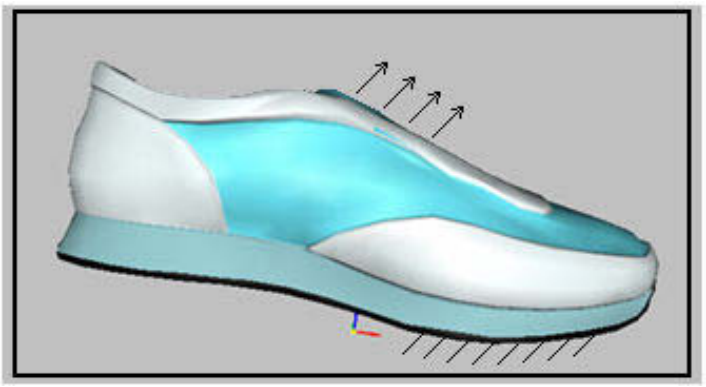
Fig. 2. Computation time with/without Straesen-Winograd algorithm.

Combining the results of the above two experiments, it can be concluded that, by adopting the local update procedure to compute inverse matrix and the SW method to compute matrix multiplication, we can improve the computational speed greatly. This proves that the accelerated approach introduced in this paper is efficient in improving the simulation performance.

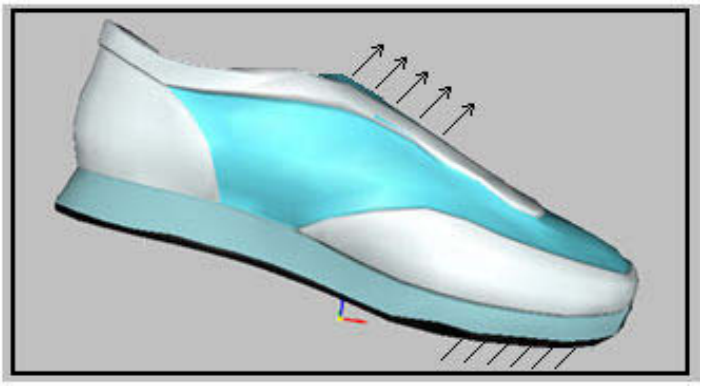
5.3 The Simulation of a Footwear Model by the Boundary Element Method
The accelerated boundary element method is implemented to simulate the deformation of some elastic models. The following figure is the application on a footwear model in different stages. Fig. 3(a). shows the original state of the sports shoe. Before the deformation, most areas of the sole bottom are in contacting with the ground. With the front portion of the sole bottom being supported by the floor, and external forces applied on the supposed contact region between the shoe and foot, the latter part of the footwear model gradually deviate from the ground. During the whole process, the contacting area between the shoe and the floor is reduced, while the touching points between the shoe and the foot is enlarged. The general deformation conforms to our intuition. Fig.3(b). and Fig. 3(c). show two stages during the deformation.



(a) footwear model before deformation



(b) footwear model during the deformation at stage 1



(c) footwear model during the deformation at stage 2

Fig. 3. The deformation of footwear model at different stages.

6. CONCLUSION

In this paper, an accelerated method which enhances the simulation efficiency on deformable bodies by BEM is presented. This improvement is led by reducing the dimension of matrices need to be inverted and applying some numerical techniques about matrix operations. Experiments and analysis prove that our method can obtain an obvious speedup on the computational time. Besides, the simulation performance is more stable by using this accelerated approach. A deformation example on a footwear model demonstrates that the deformation technique is applicable in general.

7. ACKNOWLEDGMENTS

The work described in this paper was partially supported by a grant from the Research Grants Council of the Hong Kong Special Administrative Region. (Project no. CUHK4197/04E)

8. REFERENCES

[1] Brebbia, C. A. and Dominguez, I., *Boundary Elements: An Introductory Course (Second Edition)*, Computational Mechanics Publications, c1992.

[2] Dongarra, J. J., DuCroz, J., Hanson, R. and Duff, I., A set of level 3 basic linear algebra subprograms, *ACM Trans. on Math. Soft.*, Vol. 16, No.1, 1990, pp 1-17.

[3] Douglas, C.C., Heroux, M., Slishman, G. and Smith, R.M., GEMMW: a portable level 3 BLAS Winograd variant of Strassen's matrix-matrix multiply algorithm, *Journal of Computational Physics*, Vol. 110, No. 1, 1994, pp 1-10.

[4] Dumitrescu, B., Roch, J. L., and Trystram, D., Fast matrix multiplications algorithms on MIMD architectures, *Parallel Algorithms and Applications*, Vol. 4, No. 2, 1994, pp 53–70.

[5] Duncan, W. J., Some devices for the solution of large sets of simultaneous linear equations, Philos, *Mag. Ser. 7*, 35, 1944, pp 660-670.

[6] 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.

[7] Golub, G. H. and Van Loan, C. F., *Matrix Computation*, Johns Hopkins University Press, Baltimore and London, third edition, 1996.

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

[9] Greminger, M. and Nelson, B., Deformable object tracking using the boundary element methd, *IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR2003)*, Madison, WI, June, 2003.

[10] Hager, W. W., Updating the inverse of a Matrix. *In SIAM Review*, Vol. 31, No. 2, 1989, pp 221-239.

[11] Higham, N. J., Accuracy and Stability of Numerical Algorithms, *Society for Industrial and Applied Mathematics*, Philadelphia, PA, 1996.

[12] Hui, K. C. and Leung, H.C., Virtual sculpting and deformable volume modeling, *Proceedings of Information Visualisation*, London, England, 2002, pp 664-669.

[13] Hui, K. C. and Wong, N. N., Hands on a virtually elastic object, *The Visual Computer*, Vol. 18, No. 3, 2003, pp 150-163.

[14] James, D. L. and Pai, D. K., ArtDefo: accurate real time deformable objects, *Computer Graphic*, Vol. 33, no. Annual Conference Series, 1999, pp 65-72.

[15] Sankowski, P., Dynamic transitive closure via dynamic matrix inverse (extended abstract), *45th Annual IEEE Symposium on Foundations of Computer Science (FOCS'04)*, 2004, pp. 509-517.

[16] Tang, Y. M., Zhou, A. F. and Hui, K. C., Comparison between FEM and BEM for real-time simulation, *Computer-Aided Design & Applications*, Vol. 2, No. 1-4, 2005, pp 421-430.

[17] Wunderlich, R. E., Püschel, M. and Hoe, J. C., Accelerating blocked matrix-matrix multiplication using a software-managed memory hierarchy with DMA, *Proc. High Performance Embedded Computing (HPEC)*, 2005.