Adaptively Simulating Inhomogeneous Elastic Deformation

Sei Imai[1], Yonghao Yue[1], Bing-Yu Chen[1,2] and Tomoyuki Nishita[1]
[1]*The University of Tokyo,* [2]*National Taiwan University*
etoile@nis-lab.is.s.u-tokyo.ac.jp, yonghao@k.u-tokyo.ac.jp, robin@ntu.edu.tw, nis@is.s.u-tokyo.ac.jp

Keywords: deformation, simulation, elastic object, elasticity matrix, inhomogeneous, adaptive simulation

Abstract: In this paper, we present an adaptive approach for simulating elastic deformation of homogeneous and inhomogeneous objects based on continuum mechanics. In typical adaptive simulation approaches, the deforming elastic object is usually subdivided to form a tree structure on the fly. However, they are not directly applicable for inhomogeneous elastic deformation, since the elasticity matrix, which describes the stiffness, of each element in each resolution is difficult to estimate at runtime. Furthermore, as most multi-resolution approaches, it is usually required that the stiffness of the object should either be uniform all throughout its body or consist of a collection of uniform parts, otherwise the elasticity matrices for the elements in coarse levels cannot be determined. Hence, we propose a bottom-up sampling approach to estimate the elasticity matrices for all elements in all levels based on a given stiffness function. Moreover, the subdivision process is also moved to the off-line preprocessing stage with the elasticity matrix estimation to reduce the runtime computational cost while achieving the adaptive simulation by adaptively selecting the simulation level on the fly. Therefore, we can efficiently simulate the deformation of an elastic object even with spatially varying stiffness.

1 Introduction

Physically-based simulation of soft bodies (or elastic objects), *e.g.*, jelly, is important for movies and video games. Currently a lot of techniques are carried out in soft body dynamics in computer graphics. Mass-spring- and particle-based models (*e.g.*, (Tu and Terzopoulos, 1994; Lee et al., 1995; Baraff and Witkin, 1998)) are usually used for interactive applications (*e.g.*, video games), and more accurate simulation methods based on continuum mechanics (*e.g.*, by using FEM, or finite element method) are mainly used for filming because they can usually provide higher accuracy results. Since FEM-based models require heavy calculation, several multi-resolution approaches like adaptive subdivision techniques (*e.g.*, (Debunne et al., 2001; Grinspun et al., 2002)) are proposed.

In general multi-resolution approaches, the input elastic object is divided into several elements (in our case, tetrahedra) which form a hierarchical structure. That is, several sets of elements representing the object are constructed for various resolutions (or levels), and a tree-like (*i.e.*, parent-children) relationship is established between the elements in different levels. A difficulty in using such approaches for simulating inhomogeneous elastic deformation is that the elasticity matrices, which describe the stiffness, of the elements in different levels may have different values, because a parent element may contain the children elements with different stiffness. In typical multi-resolution methods, a parent element shares the elasticity matrices with its children elements, making that the object needs to either be uniform or consist of a collection of uniform parts. Moreover, to achieve the adaptive simulation in the typical methods, the discretization (*i.e.*, tetrahedron subdivision) is usually performed on the fly. However, since the elasticity matrix of each element in each resolution is difficult to estimate at runtime, it is hard to directly apply such approaches for inhomogeneous elastic deformation.

In this paper, we present a bottom-up sampling approach to estimate the elasticity matrices for all elements in all levels based on a given stiffness function before performing the simulation. Using the presented approach, a parent element can consist of the children elements with different or even continuously varying stiffness. Moreover, the tetrahedron subdivision is also moved to the off-line preprocessing stage with the elasticity matrix estimation to ease the runtime processes while achieving the adaptive simulation by adaptively selecting the simulation level on the fly.

Our approach is composed from the off-line pre-

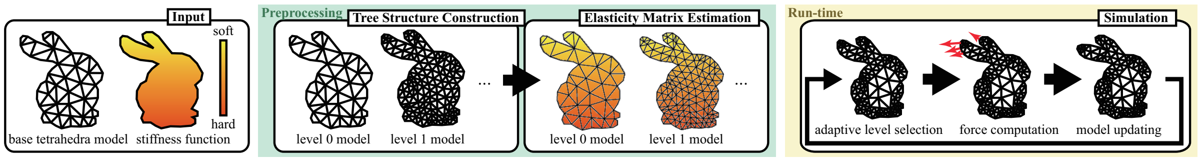

Figure 1: The overview of our adaptive simulation scheme. The illustration is drawn in 2D for simplicity.

processing and runtime stages. During the preprocessing, the input elastic object is first recursively subdivided into several tetrahedra to construct the multi-resolution tree structure while keeping the quality of the subdivided tetrahedra as good as possible for stable simulation. Next, based on the given stiffness function, the elasticity matrices of all elements (*i.e.*, tetrahedra) are estimated from fine levels to coarse levels through a sampling approach. According to the bottom-up operation, we can acquire the elasticity matrices of all elements in all levels. At runtime, the simulating elements are selected adaptively according to a user-given error-threshold and the strain posed on the elements with spatially varying stiffness. By using our approach, we can simulate the deformation of inhomogeneous as well as homogeneous elastic objects adaptively.

2 Related Work

There is a considerable amount of research on elastic deformation simulation in computer graphics (Gibson and Mirtich, 1997; Nealen et al., 2006). Methods using mass-spring- or particle-based models (Tu and Terzopoulos, 1994; Lee et al., 1995; Baraff and Witkin, 1998) are usually used for interactive applications, such as video games. To obtain more accurate results, one can use the methods that compute the continuum mechanics. Such methods can be typically classified to meshless ones (*e.g.*, (Faure et al., 2011)) and finite element methods (FEM). In this paper, we focus on FEM.

FEM divides the space into small regions utilizing finite elements like tetrahedra (O'Brien and Hodgins, 1999; Müller et al., 2001) or hexahedra (Capell et al., 2002). To improve FEM, a large amount of research has been conducted for the discretization of the input object (*e.g.*, (Shewchuk, 1998) or (Schaefer et al., 2004)), and for accelerating the computation. To reduce the heavy computation cost of FEM, adaptive subdivision approaches (*e.g.*, (Debunne et al., 2001; Grinspun et al., 2002; Dequidt et al., 2005)) are usually used. These approaches subdivide the input object according to its strain at runtime. A major problem of such methods is that they can only be applied to homogeneous objects.

There are also some methods that can simulate inhomogeneous elastic objects, *e.g.*, (Chentanez et al., 2009), but are also time-consuming. One of the speed-up approaches is numerically coarsening the linear elastic object (Kharevych et al., 2009). In this method, the inhomogeneous objects are deformed with coarsened tetrahedra, and the fine structures are mapped into the interpolated position. In (Nesme et al., 2009), the inhomogeneous elastic object is embedded into grids and the grids' elasticities are computed to achieve inhomogeneous deformation. However, to the best of our knowledge, there is no adaptive approach for simulating inhomogeneous elastic objects.

3 Adaptive Simulation Scheme

Our method is an adaptive approach and based on FEM (O'Brien and Hodgins, 1999). In typical adaptive simulation methods, the deforming elastic object is subdivided adaptively on the fly, which usually requires the material of the object to be as uniform as possible. Unlike such methods, our adaptive simulation scheme moves the subdivision process to an off-line preprocessing stage (Sec. 3.1) with the elasticity matrix estimation (Sec. 4) to reduce the runtime simulation cost (Sec. 3.2 and Sec. 5) as shown in Fig. 1. Therefore, our approach is constructed from the off-line preprocessing stage and the runtime simulation stage.

3.1 Tree Structure Construction

In our off-line preprocessing stage, we first construct the tree structure for the adaptive simulation scheme while assuming that the input object is already roughly tetrahedralized[1]. Based on this input tetrahedra set, we further recursively subdivide it to form a tree structure, while keeping the shapes of the subdivided tetrahedra as "high quality" as possible, otherwise the simulation will easily diverge at run-

[1]We believe that this assumption could be improved in the future.

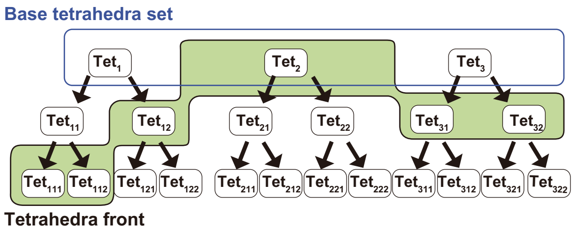

Figure 2: Illustration of the tetrahedra tree. The base tetrahedra set as the roots is the input tetrahedra set which is also the coarsest simulation level. Any cut through the intermediate tetrahedra forms a specific simulation resolution.

time. Although there are several tetrahedron subdivision approaches applicable to this tree structure construction, to guarantee the subdivided elements only consist of tetrahedra, we used a tetrahedron subdivision approach like (Liu and Joe, 1995).

3.2 Simulation with Tetrahedra Front

For a given tetrahedra tree, we first define a “tetrahedra front” as shown in Fig. 2, which is a cut through the intermediate tetrahedra to form a specific resolution for simulation and usually used in mesh simplification papers (e.g., (Hoppe, 1996)). If the “tetrahedra front” contains all leaf tetrahedra, the simulation will be performed in the finest resolution to achieve the most accurate simulation result. On the other hand, once the “tetrahedra front” only contains the “base tetrahedra set”, the simulation will be performed with only the input tetrahedra set (*i.e.*, in the coarsest level) to achieve the best simulation performance. Hence, before performing the deformation simulation, we have to first decide the “tetrahedra front” in the tetrahedra tree, in order to maximize the simulation performance while keeping the simulation accuracy to achieve the adaptive simulation (detailed in Sec. 5.2 and Sec. 5.1).

After deciding the tetrahedra front, the object can be deformed in accordance with the following equation:

$$\sigma = \mathbf{C}\varepsilon, \tag{1}$$

which is the Hooke’s law generalized to three-dimensional case, where $\sigma, \varepsilon \in \mathbb{R}^6$ is the strain and stress vectors, respectively, and $\mathbf{C} \in \mathbb{R}^{6\times 6}$ is an elasticity matrix which defines the material (*i.e.*, stiffness) of the deforming elastic object[2].

[2]In some literature, the strain, stress and elasticity are described as tensors, which are equivalent to the formulation used in this paper. We used the matrix-vector form (Zienkiewicz et al., 2005) for simplicity.

In the deformation simulation, we repeat the following process in each time-step. First, we calculate the strain vector ε of each element (tetrahedron). Next, we calculate the stress vector σ with the elasticity matrix $\mathbf{C}$ in accordance with Eq.(1). Then, we calculate the acceleration, velocity and position of each vertex at that time-step. Finally, we perform a procedure described in Sec. 5.3 to handle T-junctions.

4 Elasticity Matrix Estimation and Valid Range

After the tetrahedron subdivision, our tree structure contains only tetrahedra. If we finely subdivide the input elastic object, we can assume that all the tetrahedra in the finest level n (*i.e.*, the leaf tetrahedra) are nearly uniform, and their elasticity matrices $\mathbf{C}$ can be easily obtained from the given stiffness function[3]. Hence, we then need to estimate the elasticity matrices $\mathbf{C}$ of the elements (*i.e.*, tetrahedra) in coarse levels (*i.e.*, level 0 to $n-1$) through a bottom-up sampling approach.

Remember that an elasticity matrix $\mathbf{C}$ describes the linear relationship between a stress vector σ and a strain vector ε as shown in Eq.(1). This means that if we know the stress vector σ given the strain vector ε, we can compute the elasticity matrix $\mathbf{C}$ by solving a linear equation. Thus, the basic idea of our bottom-up approach is to first move the tetrahedra in the coarse level according to ε, simulate the deformation in the fine level, then obtain σ in the fine level and reinterpret this σ as the one in the coarse level. Finally, we can obtain $\mathbf{C}$ from the linear relationship between σ and ε in the coarse level. We describe the details below.

4.1 Strain Vector Decomposition

To make the calculation for solving the linear equation simple, we first decompose the strain vector ε into six bases as the linear combination form as:

$$\varepsilon = \sum_{m=1}^{6} \alpha^{(m)} \varepsilon^{(m)}, \tag{2}$$

where $m = 1, \ldots, 6$ is the index of each base, $\alpha^{(m)} \in \mathbb{R}$ is a scalar ranged in $-1 < \alpha^{(m)} < 1$, and $\varepsilon^{(m)}$ is one of the linear isolated strain vector bases, in which the m-th element is 1 and others are 0.

[3]If this assumption is not valid, we could apply numerical coarsening to compute the elasticity matrices

Next, since the Hooke's law is linear, we can further decompose Eq.(1) into $\sigma = \sum_{m=1}^{6} \alpha^{(m)} \mathbf{C}\varepsilon^{(m)}$, and the j-th element σ_j of the vector σ can be represented as $\sigma_j = \sum_{m=1}^{6} \alpha^{(m)} \sum_{k=1}^{6} C_{jk}\varepsilon_k^{(m)} = \sum_{m=1}^{6} \alpha^{(m)} C_{jm}$, where C_{jk} is one of the elements of $\mathbf{C}$ in the j-th column and k-th row, and $\varepsilon_k^{(m)}$ is the k-th element of $\varepsilon^{(m)}$. Alternatively, we have

$$\sigma = \sum_{m=1}^{6} \alpha^{(m)} \mathbf{C}_m, \tag{3}$$

where $\mathbf{C}_m$ is the m-th row of $\mathbf{C}$.

4.2 Elasticity Matrix Estimation

Based on Eq.(3), to determine the elements of $\mathbf{C}$, for each row m, we let $\alpha^{(m)}$ be nonzero and $\alpha^{(m')}$ $(m' \neq m)$ be zero, and compute $\mathbf{C}_m$ as $\mathbf{C}_m = \sigma/\alpha^{(m)}$. However, for accurately computing $\mathbf{C}_m$, it is not sufficient to use only a single value of $\alpha^{(m)}$ for the following two reasons. First, σ obtained from the simulation may contain numerical errors. Second, the linear relationship $\mathbf{C}_m = \sigma/\alpha^{(m)}$ is only valid for small deformations, *i.e.*, when $|\alpha^{(m)}|$ is small. Hence, we sample multiple couples of the scalar $\alpha^{(m)}$ for each $\varepsilon^{(m)}$ to estimate $\mathbf{C}_m$ as shown in Fig. 3. From the estimation, we also obtain a valid range for $\alpha^{(m)}$ such that the linear relationship is valid (detailed in Sec. 4.3).

To estimate the unknown elasticity matrix of a tetrahedron in level $n-1$ from the known elasticity matrices of tetrahedra in level n, we first, for a base m, determine a value of $\alpha^{(m)}$ and let the strain vector $\varepsilon = \alpha^{(m)}\varepsilon^{(m)}$ (*i.e.*, for other bases $m' \neq m$, $\alpha^{(m')} = 0$). Then, the vertices in level $n-1$ are moved so that the strain vector of the parent tetrahedron equals to the determined strain vector ε. From these vertices, we initialize the positions of the vertices in level n as follows. The vertices in level n can be grouped into 1) the vertices which have corresponding vertices in level $n-1$, and 2) the other vertices which do not have. For the former vertices, we fix their positions to the locations of their corresponding vertices in level $n-1$. For the latter ones, their positions are allowed to move during the process described later and are initialized using linear interpolation from the vertices in level $n-1$.

After initializing the positions of the vertices in level n, we simulate the elastic deformation of the tetrahedra in level n based on Eq.(1) using the known elasticity matrices in level n. In this simulation, only the positions of the vertices that are allowed to move are updated. After the stresses are balanced, we then calculate the stress vector σ in level $n-1$ from the stress vectors in level n to obtain one sample of the relationship between a stress vector and a strain vector as one of the blue dots shown in Fig. 3. By changing the magnitude of the scalar $\alpha^{(m)}$ arbitrarily, we can obtain multiple samples.

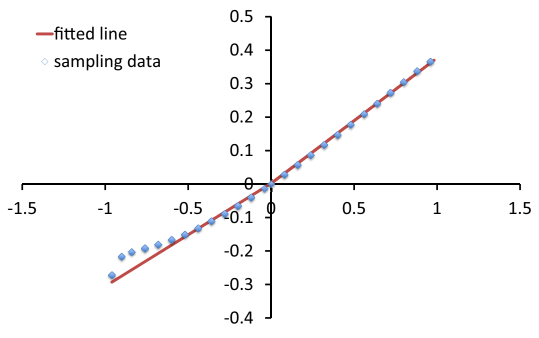

Figure 3: The relationship between $\alpha^{(m)}$ (the x-axis) and a vector component of σ (the y-axis).

Finally, we can use a straight line to fit the multiple samples while ignoring some outliers as the red line shown in Fig. 3, and the slope of the fitted line is the element of the elasticity matrix of level $n-1$.

4.3 Valid Range for Strain Vectors

As described in the previous section, the linear relationship used for estimating the elasticity matrix is only valid for small deformations. Hence, the value of the element C_{jm} of the elasticity matrix is valid if the following condition for $\alpha^{(m)}$ is satisfied.

$$\frac{|\alpha^{(m)}C_{jm} - \sigma_j(\alpha^{(m)})|}{|\alpha^{(m)}C_{jm}|} < T_1, \tag{4}$$

where T_1 is a user-specified threshold, C_{jm} is the slope of the fitted line as shown in Fig. 3, and $\sigma_j(\alpha^{(m)})$ is the value obtained from the simulation (*i.e.*, one of the blue dots in Fig. 3). Using only the above condition, however, results in erroneous conditioning when the denominator is small. Thus, we also introduce the following condition.

$$|\alpha^{(m)}C_{jm} - \sigma_j(\alpha^{(m)})| < T_2, \tag{5}$$

where T_2 is another user-specified threshold. If either of the two above conditions is satisfied, we can apply linear fitting for that range. We denote the lower and upper bounds of this valid range as $\alpha_{min}^{(m)}$ and $\alpha_{max}^{(m)}$, respectively, and set $T_1 = 1.0 \times 10^{-2}$ and $T_2 = 1.0 \times 10^{-4}$ in our experiment.

These two thresholds are the parameters to control the error bound. If they are set smaller, the allowed error becomes smaller (*i.e.*, more accurate). Thus, the valid range will be narrower and the simulation tends to switch to finer levels more often.

5 Adaptive Level Selection

Our runtime operation consists of three steps: 1) updating the tetrahedron front; 2) calculating the internal force and updating the velocity and position of each node; and 3) adjusting the vertices aligned at the T-junctions. In step 1), the tetrahedron front is updated by either replacing children tetrahedra with their parent tetrahedron, or replacing a parent tetrahedron with its children tetrahedra. We describe the conditions for these replacements in Sec. 5.1 and Sec. 5.2. For step 2), please refer to (O'Brien and Hodgins, 1999). Finally, the details of step 3) are described in Sec. 5.3.

5.1 Coarse to Fine Switching

To decide the tetrahedron front for simulation, we first check if we need to switch a tetrahedron in the tetrahedron front to its children tetrahedra in one finer level while taking into account the valid range of strain vector described in Sec. 4.3. For each tetrahedron, we calculate the strain vector ε posed on the tetrahedron and decompose it using $\varepsilon^{(m)}$ as Eq.(2). Then, if all $\alpha^{(m)}$ are in their valid ranges, *i.e.*, for all *m*,

$$\alpha^{(m)}_{min} < \alpha^{(m)} < \alpha^{(m)}_{max}, \qquad (6)$$

we regard the selected level as proper. Otherwise, we switch this tetrahedron to its children.

5.2 Fine to Coarse Switching

As using a parent (coarse) tetrahedron instead of its children tetrahedra would introduce errors into the simulation, switching from fine level to coarse level needs to be taken carefully, and indeed it is more complex. In our method, we switch to a parent tetrahedron if the following three conditions are all satisfied.

1. The strain vector of the parent tetrahedron is within the valid range as Eq.(6).
2. All of the children tetrahedra of the parent tetrahedron are contained in the tetrahedra front.
3. All strain vectors of the children tetrahedra are almost the same as that of their parent tetrahedron.

The first condition is needed to ensure that the relationship between the strain and stress vectors can be accurately handled even when we use the coarse level. The second condition is needed to avoid the case when finer levels are used to represent one or more children tetrahedra of the parent tetrahedron, as shown in Fig.4. In this case, switching to the parent tetrahedron would discard the fine structure which is needed for a sufficiently accurate calculation. The third condition is needed to take into account the actual forces posed on the children tetrahedra. If the forces largely differ across the tetrahedra, then using the parent tetrahedron would result in a crude approximation of the forces.

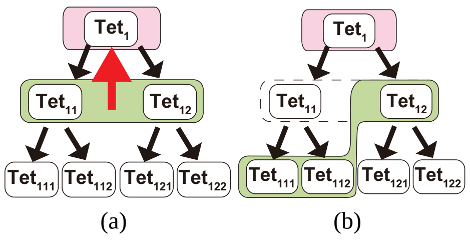

Figure 4: (a) A coarse to fine switchable case and (b) an unswitchable case. The green area is the tetrahedron front and pink area is the candidate tetrahedron to switch.

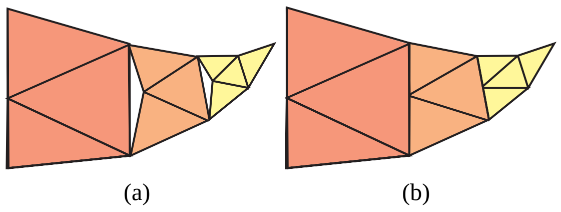

Figure 5: (a) A crack may appear due to the T-junction problem. (b) The T-junction vertex of the tetrahedra in the fine level is fixed at the middle point of its corresponding edge of the coarse tetrahedron to ensure the continuity of the object.

5.3 Handling T-junctions

Tetrahedra in different levels may cause a T-junction problem as shown in Fig. 5. At the T-junction, there is a T-junction vertex belonging to the tetrahedra in the fine level, but is not shared by their parent tetrahedron in the coarse level, a crack or overlap may thus occur. To avoid this problem, we enforce the T-junction vertex to be fixed at the middle point of its corresponding edge of the coarse tetrahedron like other adaptive modeling methods. The procedure of this enforcement is performed from the coarsest level toward the finest level.

6 Results

The simulation has been performed on a PC with an Intel Core X980 3.33 GHz CPU with 8 GB mem-

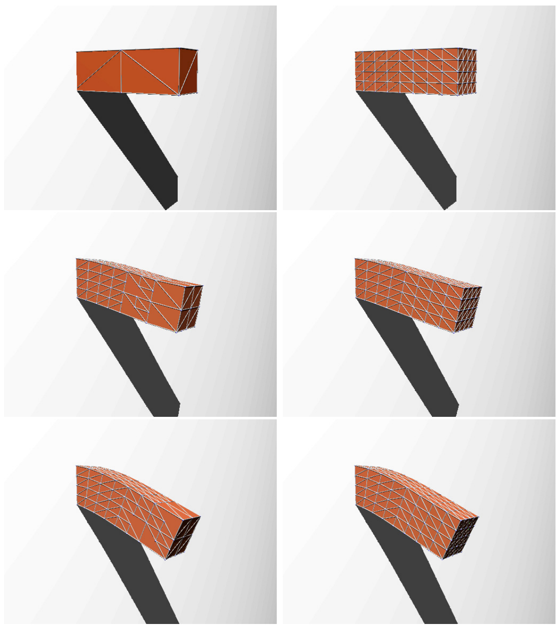
Figure 6: A comparison of our method (left) and the existing method (right).

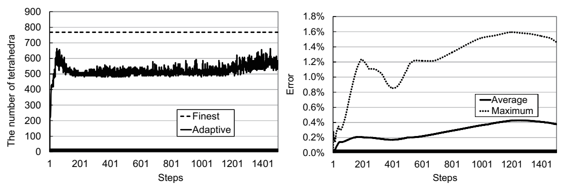

Figure 7: (a) The number of tetrahedra between our method (solid line) and the existing method (dashed line). (b) Computation error of our method. Solid line: average error. Dotted line: maximum error.

ory.

First, we show the comparison between our approach and the existing method (O'Brien and Hodgins, 1999), by simulating a deforming box, which is fixed on a wall, according to the gravity force as shown in Fig. 6. At the beginning of the simulation, the box in our approach was mostly composed of coarse tetrahedra, since the magnitude of the deformation is small for most parts (Fig. 7). A few seconds later, some parts of the box that moved a lot switched to a finer level, but the rest parts kept their original level since the movement is not so large. When the deformation reached the equilibrium, most parts were smoothly deformed and the levels they belonged to were almost the finest.

We also checked the processing time the deformation requires on each update as shown in Fig. 8. First, since most tetrahedra belonged to the coarsest level in the beginning of the simulation, the process-

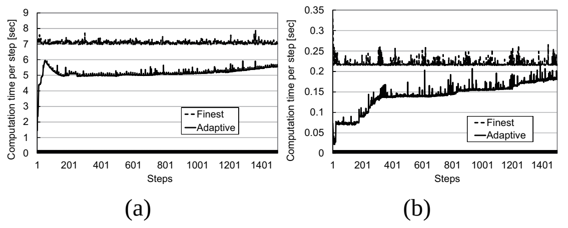

Figure 8: (a)A comparison of the calculation time with our method (solid line) and the existing method (dashed line). (b) A zoomed view.

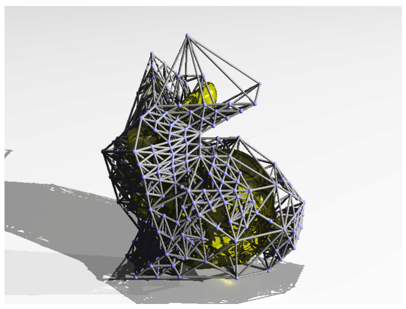
Figure 9: A smooth mesh is embedded in the tetrahedral structure.

ing time of our method was an order of magnitude faster than that of the existing method. As the simulation progressed, the level switched to a finer level and more processing time was required. When all parts switched to the finest level due to the large deformation, the calculation time was still competitive to the existing method.

The computation error between using our adaptive approach and using the finest level is shown in Fig. 7. The error is defined as the difference between the locations of the vertices when using our adaptive approach and the finest level. The error is normalized by dividing the difference by the longest edge length of the box. The average of the errors is under 0.5% and the maximum value of the errors is under 1.7%.

For rendering, we want to acquire a smooth result with the tetrahedral structure used for simulation as shown in Fig. 9. During the initial step, for each vertex of the embedded smooth mesh, we compute the corresponding finest tetrahedron the vertex belongs to, and compute the barycentric coordinates of the vertex in the tetrahedron. During the rendering step, the location of each vertex of the embedded smooth mesh is computed from a liner interpolation using the barycentric coordinates and the locations of the vertices of the corresponding tetrahedron.

To show a more complicated result, we simulated a bunny-shaped inhomogeneous jelly according to the gravity force as shown in Fig. 10. The bunny has a continuous stiffness distribution, so that the (red col-

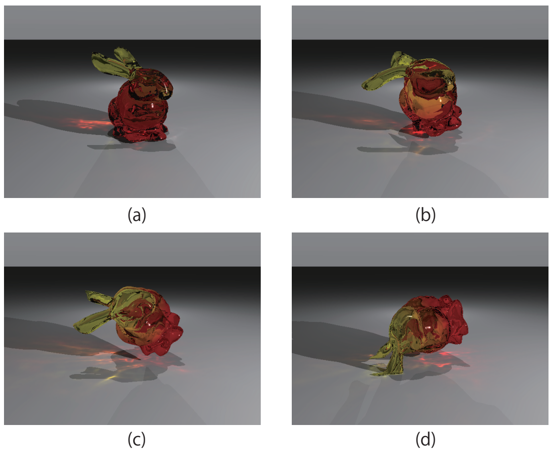

Figure 10: The simulation of a inhomogeneous bunny-shaped jelly. The stiffness of jelly is continuously different from bottom to top. The part around its feet is stiff (red) and the part around its ears is soft (yellow).

ored) bottom part is stiff but the (yellow colored) top part is soft. Because of the inhomogeneous stiffness, the bunny can stand alone thanks to its stiff feet, while its ears are hanging down from the head since they are soft.

We also simulated an armadillo-shaped inhomogeneous jelly (Fig. 12). In this example, the gravity force is not applied. The armadillo has a continuous stiffness distribution from left to right (again, red and yellow colors indicate more stiff and soft portions). We applied external forces in the same magnitude on its right and left hand. Since its right hand is more soft, with the same magnitude of external forces, it gets longer than the left hand.

To show a deformation of an object made of parts with different stiffness, we simulated the deformation of an inhomogeneous liver according to the gravity force as shown in Fig. 13. The vessels are stiffer than other tissues.

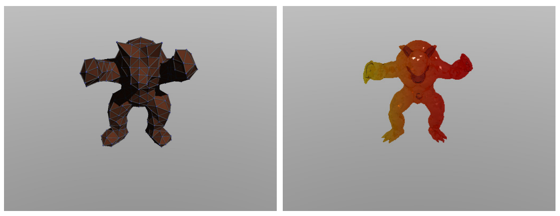

Figure 11: The simulation of an inhomogeneous armadillo-shaped jelly. The stiffness is continuously varying from left to right. We applied external forces in the same magnitude on its right and left hand.

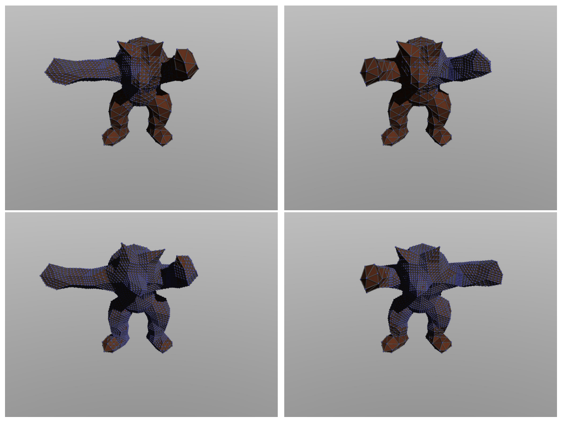

Figure 12: (left) The force is applied on its right hand. (right) The force is applied on its left hand. 400 frame passed images (top) and 800 frame passed images (bottom).

7 Conclusion and Future work

We proposed an adaptive approach for simulating elastic deformation of homogeneous and inhomogeneous objects based on FEM. A difficulty in using an adaptive as well as multi-resolution approach for simulating inhomogeneous elastic deformation is that the elasticity matrices of the elements in different levels may have different values, because a parent element may contain the children elements with different stiffness. Thus, we proposed a bottom-up sampling approach to estimate the elasticity matrices for the elements in coarse levels from the children elements with known elasticity matrices, so that a parent element can consist of the children elements with different or even continuously varying stiffness. Moreover, unlike other typical adaptive simulation methods, which subdivided the deforming object adaptively on the fly, we

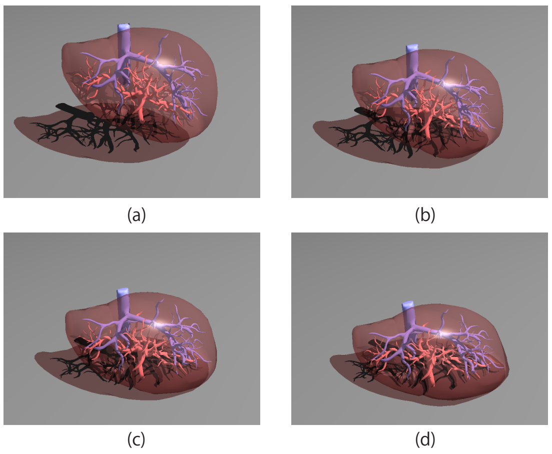

Figure 13: The simulation of liver. The vessels(blue and red) is stiffer than the other tissues.

moved the tedious tetrahedron subdivision to the offline preprocessing stage to ease the burden on the run-time simulation. Furthermore, we presented an adaptive simulation scheme, which selects the appropriate levels on the fly according to the error-threshold given by the user and the strain posed on the elements.

Our approach is faster than existing method in processing time in each time-step when coarser levels are selected. In the worst case, the processing time is still comparable to that of the existing methods. Since we can estimate the elasticity matrices in the coarse levels through a sampling approach, we believe that we can also compute the viscosity constants in a similar way.

REFERENCES

Baraff, D. and Witkin, A. (1998). Large steps in cloth simulation. In *ACM SIGGRAPH 1998 Conference Proceedings*, pages 43–54.

Capell, S., Green, S., Curless, B., Duchamp, T., and Popović, Z. (2002). A multiresolution framework for dynamic deformations. In *Proceedings of the 2002 ACM SIGGRAPH/Eurographics Symposium on Computer Animation*, pages 41–47.

Chentanez, N., Alterovitz, R., Ritchie, D., Cho, L., Hauser, K. K., Goldberg, K., Shewchuk, J. R., and O'Brien, J. F. (2009). Interactive simulation of surgical needle insertion and steering. *ACM Transactions on Graphics*, 28(3):88:1–88:10.

Debunne, G., Desbrun, M., Cani, M.-P., and Barr, A. H. (2001). Dynamic real-time deformations using space & time adaptive sampling. In *ACM SIGGRAPH 2001 Conference Proceedings*, pages 31–36.

Dequidt, J., Marchal, D., and Grisoni, L. (2005). Time-critical animation of deformable solids. *Computer Animation and Virtual Worlds*, 16(3-4):177–187.

Faure, F., Gilles, B., Bousquet, G., and Pai, D. (2011). Sparse meshless models of complex deformable solids. *ACM Transactions on Graphics*, 30(4):73:1–73:10.

Gibson, S. F. F. and Mirtich, B. (1997). A survey of deformable modeling in computer graphics. Technical Report TR97-19, Mitsubishi Electric Research Laboratories.

Grinspun, E., Krysl, P., and Schröder, P. (2002). CHARMS: a simple framework for adaptive simulation. *ACM Transactions on Graphics*, 21(3):281–290.

Hoppe, H. (1996). Progressive meshes. In *ACM SIGGRAPH 1996 Conference Proceedings*, pages 99–108.

Kharevych, L., Mullen, P., Owhadi, H., and Desbrun, M. (2009). Numerical coarsening of inhomogeneous elastic materials. *ACM Transactions on Graphics*, 28(51):51:1–51:8.

Lee, Y., Terzopoulos, D., and Waters, K. (1995). Realistic modeling for facial animation. In *ACM SIGGRAPH 1995 Conference Proceedings*, pages 55–62.

Liu, A. and Joe, B. (1995). Quality local refinement of tetrahedral meshes based on 8-subtetrahedron subdivision. *SIAM Journal on Scientific Computing*, 16(6):1269–1291.

Müller, M., McMillan, L., Dorsey, J., and Jagnow, R. (2001). Real-time simulation of deformation and fracture of stiff materials. In *Proceedings of the 2001 Eurographic Workshop on Computer Animation and Simulation*, pages 113–124.

Nealen, A., Muller, M., Keiser, R., Boxerman, E., and Carlson, M. (2006). Physically based deformable models in computer graphics. *Computer Graphics Forum*, 25(4):809–836.

Nesme, M., Kry, P. G., Jeřábková, L., and Faure, F. (2009). Preserving topology and elasticity for embedded deformable models. *ACM Transactions on Graphics*, 28(3):52:1–52:9.

O'Brien, J. F. and Hodgins, J. K. (1999). Graphical modeling and animation of brittle fracture. In *ACM SIGGRAPH 1999 Conference Proceedings*, pages 137–146.

Schaefer, S., Hakenberg, J., and Warren, J. (2004). Smooth subdivision of tetrahedral meshes. In *Proceedings of the 2004 Eurographics/ACM SIGGRAPH Symposium on Geometry processing*, pages 147–154.

Shewchuk, J. R. (1998). Tetrahedral mesh generation by Delaunay refinement. In *Proceedings of the 14th Annual Symposium on Computational Geometry*, pages 86–95.

Tu, X. and Terzopoulos, D. (1994). Artificial fishes: physics, locomotion, perception, behavior. In *ACM SIGGRAPH 1994 Conference Proceedings*, pages 43–50.

Zienkiewicz, O., Taylor, R., Taylor, R., and Zhu, J. (2005). *The finite element method: its basis and fundamentals*. The Finite Element Method. Elsevier Butterworth-Heinemann.