Real-Time Simulation of Flexible Materials in Avango Virtual Environment Framework

Stanislav Klimenko[(1)], Lialia Nikitina[(2)], Igor Nikitin[(2)]
[(1)]*Institute of Computing for Physics and Technology, Protvino, Russia*
[(2)]*Fraunhofer Institute for Media Communication, Sankt Augustin, Germany*

Abstract. *This paper describes a system for real-time simulation of linear elastic deformations of volumetric objects in Virtual Environments, implemented in Avango VE Framework* [1]. *The approach makes use of the methods of Finite and Boundary Elements and precomputed Green's functions, and supports real-time interactive deformations of large elastic models, containing more than 10,000 volumetric nodes, at the graphics update rate 85fps.*

1. Introduction

Simulation of an object's elastic deformation is an important feature in applications where three-dimensional object behavior is explored. In addition, the benefits of user-object interactions are best realized in interactive environments which require the rapid computation of deformations. The accurate physically-based methods for simulation of elastic deformations are required especially in engineering and medical applications. Pioneering work in this area has been done by Bro-Nielsen and Cotin [2] using the Methods of Finite Elements to simulate elastic deformations. The survey paper[3] describes much of the previous work on the modeling of deformable objects. Recent advances are the application of the Boundary Element Method and fast update Sherman-Morrison-Woodbury algorithm for the simulation of linearly elastic homogeneous objects[4], and implementation of St. Venant-Kirchhoff materials[6] and modified nested dissection[7] for the simulation of non-linear elasticity. The existing approaches to simulation of elastic deformations differ by implementation of the following key issues.

Linear vs non-linear. Most of the approaches[2, 4, 8, 9] use linearized elastic equations (so called theory of small deformations), due to their relative simplicity and accelerated computational performance comparing with the non-linear case. There are two types of non-linearity, which can complicate the simulation: material and geometrical ones[10]. The material non-linearity, caused by violation of stress-strain linear relation (Hooke's law), in engineering applications usually occurs only under extremal conditions when the material looses the elastic properties, while in bio-mechanics the material non-linearity is often encountered[11]. The geometrical non-linearity occurs at large displacements of points in the body relative to each other and is associated with a non-linear term of purely geometrical nature, appearing in the strain tensor. The contribution of this term depends on the geometrical shape of the object. For objects that have approximately equal length in all three dimensions (such as sphere or cube), the large relative displacements can appear only as a result of unrealistically large stress. For practical cases the geometrical non-linearities are negligible, and the theory of small deformations can be applied for this kind of bodies[10]. In the contrast, for the deformations of long tubes and cylindric bending of thin plates the large displacements are possible without creation of much stress. The theory of small deformations is still applicable for this kind of bodies if the displacements are artificially restricted to small values. Beyond these limits one needs to apply either general non-linear methods[6, 7, 12] or specialized approximations[10] for tubes and plates.

Precomputation vs on-line solution. For linear problems, there is the possibility to solve the equations of elasticity on-line and simultaneously perform the graphical rendering, or to perform some of these computations off-line. Linear problems in theory of elasticity require a solution of large linear systems of the form $Ku = f$ with constant matrix K and variable right hand side f. Using on-line methods, one can achieve the real-time performance for moderately large models, typical figures are given in[6]: 1,400 nodes at 45 fps for PC Pentium III 500 MHz. The elements of inverse matrix: $(K^{-1})_{ij}$ describe how the influence of the unit force applied to node j propagates through the body to node i. In continuous limit it coincides with the Green's function $G(x,y)$ which describes the propagation of the influence from point x to point y. Using off-line inversion of matrix K and representation of solution as $u = K^{-1}f$, a better performance can be obtained[14]: 10,000 nodes at 85 fps for PC Athlon 1.3 GHz. The advantage of on-line approach is a possibility to solve non-linear problems, according to[6], this can be done at nearly real-time speed: 1,400 nodes at 8 fps. One more property, which is usually considered as an advantage of on-line approach, is a possibility to perform interactive change of system matrix K, including those associated with variation of boundary conditions and change of topology (cuts) of the model. However, in recent works[4, 13] these features have been implemented for precomputed models as well, using a fast update algorithm for evaluation of K^{-1}.

Quasistatic vs dynamical. Quasistatic scheme[4, 14] each frame finds the exact equilibrium of the object with respect to the given interaction. In dynamical scheme[6, 7] the elastic forces acting at the nodes of the model are found and further evolution governed by Newton's laws is computed in real-time with standard integration methods, modeling properly the masses and damping forces distributions. Although the dynamical scheme is able to represent physically correct evolution, the simulated relaxation processes are usually slower than in physical reality, especially for large models, due to insufficient computational power and restrictions imposed on time step by stability of integration methods.

FEM vs BEM. Finite Element Method[15] subdivides the body to a finite set of primitives, using e.g. tetrahedral mesh, with subsequent definition of a physical equilibrium for each of these elements. Boundary Element Method[4] uses analytical reformulation of original partial differential equations in the theory of elasticity to an integral form, which includes only the surface variables (displacements and tractions). In BEM only the surface of the object should be meshed, practically one can take the same triangulation as used for the rendering. In FEM the interior also should be meshed, even in the case if it should not be visualized. The application of BEM is restricted to the objects with homogeneous interior. The objects with complex internal structure can be processed only by FEM. Linear systems, generated by FEM, are large and sparse, while the equivalent BEM systems are small and dense. Both methods can be used in pre-computation mode to produce the Green's functions, which can then be passed on to the on-line part of the simulator.

Interaction mode: positional vs force. For interaction with deformable model the most straightforward approach[2] is to specify the force, acting at separate nodes or distributed on the surface. The other possibility is to specify directly the displacements for some of the nodes. This approach is used e.g. in modeling of the contact between elastic and rigid objects, in this case more complex variable boundary condition problem should be solved[4]. During this solution the positional input is internally converted to force one, by evaluating the contact force distribution in terms of displacements. Haptic simulation [8,9] requires also fast computation of the output force to support high update rate necessary for haptic devices.

Interaction scheme: low-dimensional vs high-dimensional. In special interaction schemes, when the interaction with the object is performed via a small set of the control elements attached to its surface[14], the object effectively receives the same number of degrees of freedom as contained in the control elements. In this case the model is deformed in restricted low-dimensional interaction space, where the data size and computational load can be significantly reduced. For the interaction schemes, where the user can interact with an arbitrary group of nodes in the model, e.g. in the contact problems, complete information about the system response is needed. In this case larger data volumes should be processed, allowing only the partial reduction of the problem to visible surface nodes[2,4].

This paper presents a system for real-time simulation of elastic deformations implemented by us in *Avango* Virtual Environment Framework[1]. The approach is based on precomputation of Green's functions for the theory of elasticity in linear quasistatic formulation. The system supports FEM- and BEM-precomputation of large models, and their real-time deformations using positional and force input for low-dimensional and high-dimensional interaction schemes. The rest of this paper is organized as follows. In the second section we present the implementation of flexible materials in Avango, focusing at the optimization techniques used to accelerate the simulation and new features extending its capabilities. In the third and fourth sections the obtained results are summarized.

2. Implementation of flexible materials in Avango VE framework

Avango[1] is a programming framework for building distributed, interactive VE applications. It is based on OpenGL/Iris Performer [16] to achieve the maximum possible performance for an application and addresses the special needs involved in application development for virtual environments. Avango uses the C++ programming language to define the objects and scripting language Scheme[17] to assemble them in a scene graph. Avango objects contain state information in a collection of *fields*. Connections between fields form a *dataflow graph*, conceptually orthogonal to the scene graph, which is used to define interaction between the objects and to import the data from external devices into the application.

We have implemented flexible materials simulator as a component of Avango system. The simulation uses the following general scheme (fig.1). At the first step the user-provided input data are pre-processed in off-line mode. This is most time-consuming part of the computational process, which can take minutes or hours dependently on complexity and type of the model. During this stage pre-computation of a complete basis of solutions (Green's functions) of the system is performed. The resulting data are saved to a file and used to accelerate the interactive visualization module, operating in Virtual Environments at real-time speeds.

2.1. Input data

FEM-based scheme requires the information about the shape of object, its volumetric tetrahedral subdivision, representation of the surface in the form of triangle stripsets, data on internal composition of the object (material constants: mass density ρ, Young's module Y, Poisson's ratio ν) and definition of control elements for the case of low-dimensional interaction scheme. These data are provided in a file with the structure, shown in Table 1. For the objects of special type, such as tubes and cables, these data can be generated from a small set of user-defined parameters, see Table 2.

Table 1: input data, general case

values	type [units]
numbers of nodes, tetrahedrons, strips	integers
coordinates of nodes	triples of floats, [m]
indices of nodes in tetrahedrons	quadruples of integers
material constants, per tetrahedron: ρ, Y, ν	triples of floats [kg/m^3, N/m^2, –]
lengths of strips, per strip	integer
indices of nodes in strips	list of integers
control element membership flag, per node	character

Table 2: input data for tubes and cables

values	type [units]
length of the tube	float, [m]
external and internal diameters	floats, [m]
width of control element (default: external diameter)	float, [m]
predeformed curvature c, to support tubes, whose initial shape is circular arc; inverse to the radius of curvature $c = 1/r$, for initially straight tubes $c = 0$	float, [1/m]
material constants ρ, Y, ν	floats, [kg/m^3, N/m^2, –]
numbers of subdivisions of volumetric grid in axial, azimuthal and radial directions	integers

BEM-scheme is applicable for objects with homogeneous interior, and requires the definition of material constants (uniform for the whole object) and surface geometry, represented in the form of triangle stripsets, using e.g. standard Open Inventor format. Topologically the surface should be smooth closed connected embedding (boundary of the region). Its tessellation should not have singularities, such as degenerate triangles, and should be waterproof: holes or overlaps, even microscopic, are not allowed. In our simulator the last property is controlled by topological “hedgehog” theorem (total vector area of tessellation should vanish).

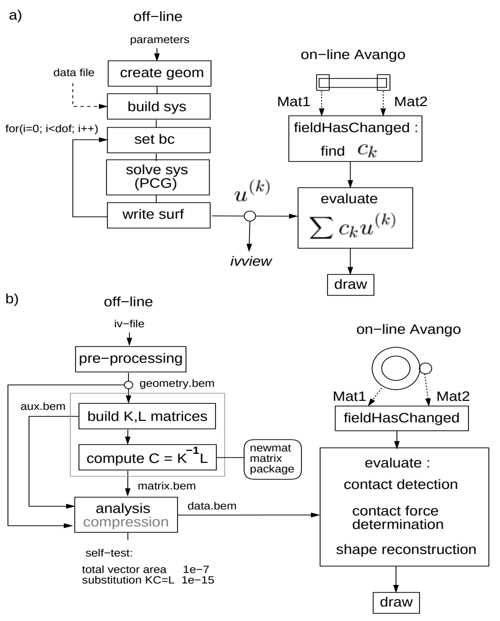

Fig.1. Simulation schemes: (a) using FEM and low-dimensional interaction; (b) using BEM and high-dimensional interaction.

2.2. Pre-processing

Pre-processing starts from building of the linear system, describing the physical equilibrium of the model, which has a form $Ku = f$, where f are external forces, acting at the nodes of the model, u are unknown displacements of the nodes, K is a large symmetric matrix (called stiffness matrix). In the case of FEM this matrix is sparse, containing typically more than 99% of zeros. For BEM analogous system can be written: $Ku = Lp$, where p is surface density of external force (also called traction), estimated at a given node of the surface, u are displacements of the surface nodes, K and L are dense matrices. The appropriate definitions of the matrix elements can be found in[15, 2] for FEM and in[4] for BEM.

The main part of the pre-processing stage is the inversion of system matrices and representation of solutions in the form $u = K^{-1}f$ or $u = K^{-1}Lp$. Before that an additional boundary condition should be imposed to the system, removing its degeneration with respect to rigid motions (translations and infinitesimal rotations). This boundary condition fixes a part of the object in a certain coordinate frame and remains permanently imposed during further deformations. It is equivalent to removal of the displacements for fixed vertices in the linear system. This procedure makes the matrix K non-degenerate, therefore we are able to invert it. For dense matrices of BEM case we use LU-decomposition algorithm, implemented in general purpose C++ matrix library *newmat*[18]. For sparse FEM matrices we use preconditioned conjugate gradient method PCG[19] together with a scheme [14] for compact storage of matrix K. This scheme stores only non-zero entries of matrix K, using data structures related to the mesh itself: the diagonal elements K_{ii} are stored in the nodes of the mesh, while the non-diagonal elements K_{ij} are stored on the edges.

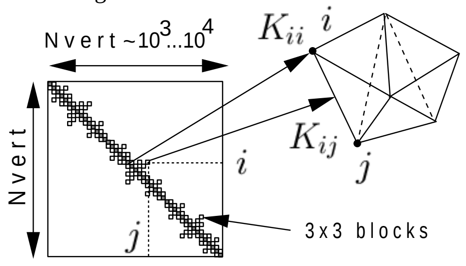

Fig.2. Storage of stiffness matrix.

In the case of positional input we use block decomposition of matrix K (called also condensation technique[2, 8]), writing $K_{nn}u_n = -K_{nb}u_b + mg$, where index b is used for those nodes, whose positions are interactively specified by user (e.g. by means of the control elements), while index n is used for other nodes, whose positions are unknown. The term mg represents a column $(m_1\vec{g}, m_2\vec{g}...)^T$, where $\vec{g}$ is gravitational field and m_i is effective mass of the node, equal to the quarter-sum of masses for all tetrahedrons adjoint to the node. Therefore the answer can be written as $u_n = -(K_{nn}^{-1}K_{nb})u_b + K_{nn}^{-1}mg$. Further simplification is possible in usage of low-dimensional interaction scheme[14], where the displacements u_b are given by affine transformations $\vec{u}_b = \vec{T} + M\vec{u}_{b0}$. Each transformation is associated with a control element attached to the elastic object. It is described by 12 parameters, namely 3 for the translations $\vec{T}$ and a 3×3 matrix of general linear transformation M (including 3 rotations and 3 scalings in particular), defining a 12-dimensional space per a control element. Let $u_b^{(k)}$, $k = 1..12$ be a basis in this space, corresponding to the unit placed sequentially to the entries of $\vec{T}$ and M, while other entries are filled by zeros. Let $u_n^{(k)}$ be corresponding solutions of the system, i.e. $u_n^{(k)} = -K_{nn}^{-1}K_{nb}u_b^{(k)}$. Such solutions should be found for each of N control elements. Three additional solutions should be also considered $u_n^{(12N+i)} = K_{nn}^{-1}mg_i$, $i = 1, 2, 3$, corresponding to three possible directions of gravity vector $\vec{g}$. Then, due to linearity of the system, the shape of the elastic body for a given interaction $u_b = \sum_k c_k u_b^{(k)}$ and gravity vector $\vec{g} = \sum_i c_{12N+i}\vec{g}_i$ will be

$$u_n = \sum_{k=1}^{12N+3} c_k u_n^{(k)}.$$

In the obtained solutions only the entries u_s, corresponding to visible surface nodes should be stored. Actually, we pre-compute solutions for elementary interactions modes, corresponding to the basis vectors $u_b^{(k)}$ and three directions of gravity, in the off-line mode using PCG. The result is saved (only for surface nodes) to Open Inventor file, see fig.3. Here white color marks non-deformed shape, yellow – solutions for $\vec{T}$-component of two affine transformations, corresponding to two control elements, red – solutions for M-component, green – gravitational modes. In visualization module these solutions are linearly combined to find the shape of the object for any given interaction using fast on-line computation.

Note: similar approaches can be used to support the haptic simulation[8, 9]. From the second part of K-matrix block decomposition one can write $K_{bn}u_n = -K_{bb}u_b + mg + f_b$, where f_b is an external force, acting at b-nodes. After substitution of u_n in terms of u_b and summation $f = \sum_b f_b$ one can obtain a smaller system, whose solutions can be also precomputed and used for haptic simulation. In the contrast to visual system, where the solution should be evaluated in each node of the surface, the haptic system requires the evaluation of a single vector of the force. Practically this allows to compute the haptic force at much higher update rate than solution of visual system, achieving the performance necessary for haptic devices (>1kHz). In this case the haptic simulation part should work in a separate thread[9] or even as a master application, transmitting the input data to the visualization module.

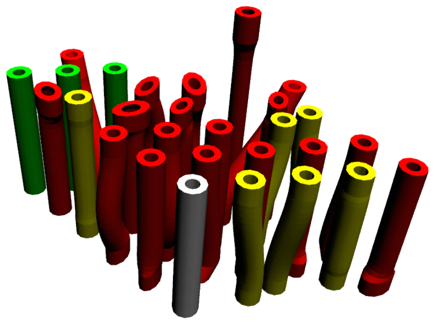

Fig.3. Precomputed basis of solutions in Open Inventor file.

2.3. On-line visualization module

Interaction with virtual objects in Avango is supported by a concept of *draggers*. The dragger attached to an object detects on a low level the intersection of the interaction device with the object and copies the device's matrix to the object's matrix. This allows easy positioning of virtual objects. We use this feature in our simulator as follows. The draggers are attached to the control elements in low-dimensional interaction scheme and to the contacting objects in high-dimensional one, then their matrices are submitted to the visualization module using field connections, see fig.1. Whenever matrices are changed, the method *fieldHasChanged* is activated, which extracts the necessary data from the entries of the matrices. Actual computation is postponed to the *evaluate* method, activated only once per frame if the field has been changed. The main functionality of the simulator is encapsulated to C++ classes, e.g. for low-dimensional interaction scheme:

```
class Deform{
  public:
    int load(const char* filename);
    int evaluate();
    pfMatrix *Mat; ushort N;
    pfVec3 *Verts, *Norms, *Colors;
    ulong numVerts,numStrips,
          *stripLengths,*strips;
    float criticalStrain,
          criticalStretch, criticalAngle;
};
```

The method *load*() reads data from specified Open Inventor file, returning 1 on successful loading, 0 otherwise. It (re-)allocates and initializes the arrays `Mat`, `Verts`, `Norms`, `Colors`, whose content is updated during the simulation, and the arrays for tristrip data, remaining unchanged.

The method *evaluate*() computes for the given matrices of control elements `Mat[]` the shape of the object's surface, its normals, and color representation of deformation. The main part is evaluation of the sum $\sum_k c_k u^{(k)}$ with the basis shapes $u^{(k)}$ taken from the data file and the coefficients c_k extracted from the entries of the input matrices. The *evaluate*() returns 1, if solution satisfies a condition of small deformations (linearity test), and returns 0 otherwise. In the last case the interaction is blocked and the visualization arrays are left unchanged.

Linearity test compares strain- or stress-based characteristic of the deformation (described below) with the user provided threshold value, e.g. `criticalStrain`. Additionally, for the case of thin plates and long tubes, where nonlinear deformations can appear without large strain or stress, the user can provide threshold values for the displacements and rotation angles of the control elements, thus prohibiting large deformations. To support wider linearity region, we have implemented the approach proposed in paper[14]: the average rotation matrix of the control elements is extracted and transferred to the rigid rotation of the whole model. This allows to reduce the rotation angles of control elements relative to the average position, significantly decreasing nonlinear terms, proportional to the angles squared. The average rotation matrix is defined as $R = GS(\sum R_i/N)$, where R_i are rotation matrices of control elements relative to their default positions and $GS()$ is Gram-Schmidt orthonormalization procedure. For large rotation angles this procedure has singularities, particularly, in the case of two control elements it is singular for matrices satisfying a relation $R_2 = R_\pi R_1$, i.e. matrices, related by a rotation by the angle π about some axis. Such singularities do not appear in the simulation, because large relative rotation angles correspond to large deformations, prohibited by the linearity test. To support correct transformations of the model in fixed gravity field, while the model is subjected to the extracted average rotation the effective gravity vector should be transformed by the inverse one: $\vec{x}(\vec{g}) \to R\vec{x}(R^T\vec{g})$. Namely for this purpose the precomputation of three gravitational modes is necessary, even in the case if the direction of gravity vector is fixed.

Characterization of deformations by color is convenient to detect the regions of the object, subjected to strong deformations, and to intensify visual force feedback. For characterization of the deformations one can use the components of strain or stress tensors, or their functions. Surface components of strain tensor are directly computable in terms of deformation of the object's surface. Particularly, the trace of surface strain tensor is equal to relative change of area for given triangle in surface tessellation: $\Delta S/S = \varepsilon_{11} + \varepsilon_{22}$. This characteristic is especially simple for the evaluation. Analogously, the trace of complete strain tensor is equal to the relative change of volume for given tetrahedron: $\Delta V/V = \varepsilon_{11} + \varepsilon_{22} + \varepsilon_{33}$. Here the third axis is directed along the normal to the surface. Note that volumes of tetrahedrons are not accessible in our representation of the result, where only the surface geometry is present. The normal components of strain tensor contain normal derivatives, which also cannot be expressed in terms of surface deformation. Normal derivatives enter also to stress tensor and such characteristics as distribution of elastic energy, complicating their evaluation. One approach, applicable for low-dimensional interaction scheme, is to use linearity of strain and stress tensors in terms of interaction coefficients c_k and precompute corresponding basis functions. This approach requires more data to store and more computations to perform on-line. The other approach is to compute the normal derivatives on-line using boundary conditions satisfied on the free surface [4].

3. Results

Fig.4 presents sample flexible objects, deformed by our simulator. Models (a-d) were pre-processed using FEM for low-dimensional interaction scheme, while the model (e) was computed by BEM for high-dimensional interaction scheme with contact detection. The table below shows parameters of the models, required pre-processing time (for HP 2GHz Linux PC) and the size of compressed output file:

Table 3: parameters of the precomputed models

model	tets	volume nodes	surface nodes	comp. time	file size
(a)	55296	13968	9408	18 min	2.0Mb
(b)	57600	14472	9696	2h15m	2.3Mb
(c)	69120	14520	3096	7h11m	0.9Mb
(d)	26422	5292	3510	1h24m	1.0Mb
(e)	–	–	800	30 min	10Mb

The interactive deformations of these models are performed in the virtual environment by on-line visualization module, working stable at graphics update rate 85fps (on HP 2GHz Linux PC). In the case (a) one end of a polyethylenic tube is pulled on a rigid connector, using scaling of the corresponding control element; wireframe mode is used to show the detalization level of the model. In the case (b) a rubber tube sags in gravity field under its own weight. The case (c) shows a flexible cable initially curved to a circle and interactively deformed to a spring shape, using two control elements attached to the ends. In the case (d) a composite object, consisting of steel plate and rubber block, is deformed by two control elements. In the cases (c,d) color represents the trace of surface strain tensor for the deformation. In the case (e) a rigid sphere contacts the elastic object and can be interactively moved along its surface; color represents the distribution of elastic energy in the object.

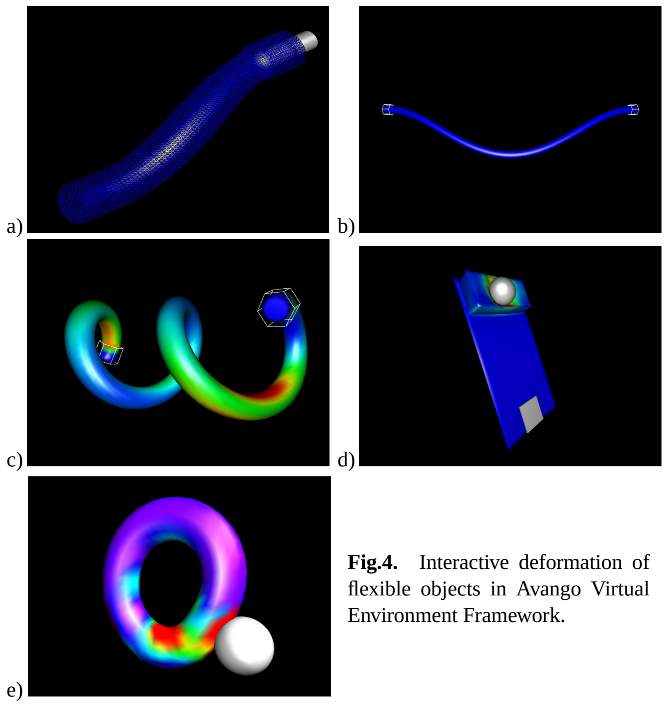

Fig.4. Interactive deformation of flexible objects in Avango Virtual Environment Framework.

4. Conclusion

In this paper we described our implementation of real-time simulation of elastic deformable objects in *Avango* VE Framework. The approach uses the finite and boundary element methods to solve the equations of elasticity. The most time consuming portions of the computations are performed in off-line mode. The resulting data, saved to a file, allows to accelerate significantly the on-line simulation process. We have implemented this approach in the simulator of flexible materials, supporting interactive deformations of large elastic models in virtual environment at real-time speeds.

Acknowledgments. This work has been partially supported by RFBR grants: projects 02-01-01139 and 01-07-90327.

References

1. Tramberend H., Avocado: A Distributed Virtual Reality Framework, Proc. of the IEEE Virtual Reality, 1999.
2. Bro-Nielsen M., Cotin S., Real-Time Volumetric Deformable Models for Surgery Simulation using Finite Elements and Condensation, Eurographics'96, Vol.15 (1996) No.3, C57-C66.
3. Gibson S.F., Mirtich B., A survey of deformable models in computer graphics, Technical Report TR-97-18, Mitsubishi Electric Research Laboratories, Cambridge, MA, November 1997.
4. James D., Pai D., Artdefo Accurate Real Time Deformable Objects, Computer Graphics, vol.33, pp.65-72, 1999.
5. James D., Pai D., A Unified Treatment of Elastostatic Contact Simulation for Real Time Haptics, Haptics-e, vol.2, num.1, September 27, 2001.
6. Picinbono G., Delingette H., Nicholas H.-A., Non-Linear Anisotropic Elasticity for Real-Time Surgery Simulation, INRIA Research Report 4028, October 2000.
7. Y.Zhuang, J.F.Canny, Real-time global deformations, In Algorithmic and Computational Robotics, pp.97-107, Natick MA, 2001. A.K.Peters.
8. A. O. Frank, I. A. Twombly, J. D. Smith, Finite Element Methods for real-time Haptic Feedback of Soft-Tissue Models in Virtual Reality Simulators, Proceedings of VR2001.
9. J. Berkley et al, Banded Matrix Approach to Finite Element Modeling for Soft Tissue Simulation, Virtual Reality: Research, Development, and Applications (1999) 4:203-212.
10. L. D. Landau, E. M. Lifschitz, Theory of Elasticity, volume VII of Theoretical Physics, Moscow, Nauka, 1970.
11. W.J. Niessen, M.A. Viergever (Eds.): Medical Image Computing and Computer-Assisted Intervention - MICCAI 2001, 4th International Conference, Utrecht, The Netherlands, October 14-17, 2001, Proceedings. Lecture Notes in Computer Science 2208, Springer 2001.
12. Hirota, G., Fisher, S. and Lin, M., Simulation of Non-penetrating Elastic Bodies Using Distance Fields. UNC Technical Report TR00-018, 2000.
13. U. Meier, C. Monserrat, N.-C. Parr, F.J. Garcia, J.A. Gil, Real-Time Simulation of Minimally-Invasive Surgery with Cutting Based on Boundary Element Methods, Lecture Notes in Computer Science (2001) V.2208, p.1263.
14. I. Nikitin, L. Nikitina, P. Frolov, G. Goebbels, M. Goebel, S. Klimenko, G.M. Nielson, Real-time simulation of elastic objects in Virtual Environments using finite element methods and precomputed Green's functions, Proc. of EGVE 2002 (Barcelona, Spain, May 2002), p.47, published by ACM, New York, 2002.
15. O. C. Zienkiewcz, R. L. Taylor, The Finite Element Method, Vol.1, Edition 5, Butterworth-Heinemann, Oxford 2000.
16. J.Rohlf and J.Helman. IRIS Performer: A High Perfomance Multiprocessing Toolkit for Real Time 3D Graphic. In A. Glassner, editor, Proceedings of SIGGRAPH '94, pp. 381-395.
17. R. Kent Dybvig. The Scheme programming language: ANSI Scheme. P T R Prentice-Hall, Englewood Cliffs, NJ 07632, USA, Edition 2, 1996.
18. R. Davies, NewMat C++ Matrix Class, `http://ideas.uqam.ca/ideas/data/Softwares/codccplusnewmat.html`
19. W.Press, S.Teukolsky, W.Vetterling, and B.Flannery. Numerical Recipes in C. Cambrige, 1994.
20. S.Wolfram, The Mathematica^R Book, Cambridge University Press 1999.
21. M.Lin, A.Gottschalk, Collision Detection between Geometrical Models: A Survey, Proc. IMA Conference on Mathematics of Surfaces, 1998.