SOFA – an Open Source Framework for Medical Simulation

J. ALLARD [a] S. COTIN [a] F. FAURE [b] P.-J. BENSOUSSAN [b] F. POYER [b]
C. DURIEZ [b] H. DELINGETTE [b] and L. GRISONI [b]
[a] *CIMIT Sim Group - Harvard Medical School*
[b] *INRIA - Evasion, Alcove, and Asclepios teams*

Abstract. SOFA is a new open source framework primarily targeted at medical simulation research. Based on an advanced software architecture, it allows to (1) create complex and evolving simulations by combining new algorithms with algorithms already included in SOFA; (2) modify most parameters of the simulation – deformable behavior, surface representation, solver, constraints, collision algorithm, etc. – by simply editing an XML file; (3) build complex models from simpler ones using a scene-graph description; (4) efficiently simulate the dynamics of interacting objects using abstract equation solvers; and (5) reuse and easily compare a variety of available methods. In this paper we highlight the key concepts of the SOFA architecture and illustrate its potential through a series of examples.

1. Introduction

Computer-based training systems offer an elegant solution to the current need for better training in Medicine, since realistic and configurable training environments can be created. This can bridge the gap between basic training and performing the actual intervention on patients, without any restriction for repetitive training. However, in spite of the impressive developments in the field of medical simulation, some fundamental problems still hinder the acceptance of this valuable technology in daily clinical practice. In particular, the multi-disciplinary aspect of medical simulation requires the integration, within a single environment, of leading-edge solutions in areas as diverse as visualization, biomechanical modeling, haptics or contact modeling. This diversity of problems makes it challenging for researchers to make progress in specific areas, and leads rather often to duplication of efforts.

1.1. Objectives

For the past few years, there have been a few attempts at designing software toolkits for medical simulation. Examples include SPRING [7], GiPSi [3], VRASS [4], or SSTML [1]. These different solutions aim at the same goal: providing an open source answer to the various challenges of medical simulation research and development. Although our aim is identical, we propose a different approach, through a very modular and flexible software framework called SOFA. This open source framework allows independently developed algorithms to interact together within a common simulation while minimizing the development time required for integration.

The main objectives of the SOFA framework are:

- Provide a common software framework for the medical simulation community
- Enable component sharing / exchange and reduce development time
- Promote collaboration among research groups
- Enable validation and comparison of new algorithms
- Help standardize the description of anatomical and biomechanical datasets

Our main overall goal is to develop a flexible framework while minimizing the impact of this flexibility on the computation overhead. To achieve these objectives, we have developed a new architecture that implements a series of concepts described below.

2. The SOFA architecture

The SOFA architecture relies on several innovative concepts, in particular the notion of **multi-model representation**. In SOFA, most simulation components – deformable models, collision models, instruments, etc – can have several representations, connected together through a mechanism called mapping. Each representation can then be optimized for a particular task – e.g. collision detection, visualization – while at the same time improving interoperability by creating a clear separation between the functional aspects of the simulation components. As a consequence, it is possible to have models of very different nature interact together, for instance rigid bodies, deformable objects, and fluids. At a finer level of granularity, we also propose a **decomposition of physical models** – i.e. any model that behaves according to the laws of physics – **into a set of basic components**. This decomposition leads for instance to a representation of mechanical models as a set of degrees of freedom and force fields acting on these degrees of freedom. Another key aspect of SOFA is the **use of a scene-graph to organize and process the elements of a simulation** while clearly separating the computation tasks from their possibly parallel scheduling. These concepts not only characterize SOFA but also provide a mean to address the goals described in section 1.1.

2.1. High-Level Modularity

Any simulation involves, to some extent, the computation of visual feedback, haptic feedback, and interactions between medical devices and anatomical structures. This typically translates into a simulation loop where, at each time step, collisions between objects are detected, deformation and collision response are computed, and the resulting state can be visually and haptically rendered. To perform each of these actions, the various algorithms involved in the simulation rely implicitly on different data structures for the simulated objects. In SOFA we explicitly decompose an object into various representations, in such a way that each representation is more suited toward a particular task – rendering, deformation, or collision detection. Then, these representations are linked together so they can be coherently updated. We call the link between these representations a *mapping*. Various mapping functions can be defined, and each mapping will associate a set of primitives of a representation to a set of primitives in the other representation (see Figure 1). For instance, a mapping can connect degrees of freedom in a Behavior Model to vertices in a Visual Model.

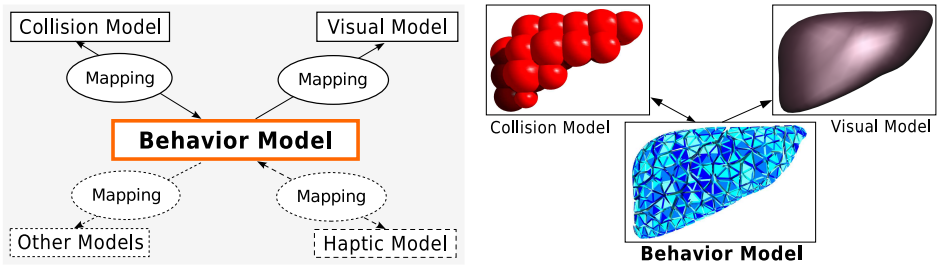

Figure 1. Illustration of the multi-model representation in SOFA. *Left*: possible representations for a simulated object, with the Behavior Model controlling the update of the other representations through a series of mappings. *Right*: examples of these representations for a liver model. Notice how the Visual Model is more detailed than the Behavior Model and how the Collision Model relies on a very different representation.

2.2. Fine Grain Modularity

One of the most challenging aspect of medical simulation is the computation, in real-time, of accurate biomechanical models of soft-tissues. Such models being computationally expensive, many strategies have been used to improve computation times or to reduce the complexity of the original model: linear elastic models have often been used instead of more complex non-linear representations, mass-spring methods as an alternative to finite element methods, etc. Each of these simplifications induces drawbacks, yet the importance of these drawbacks depends largely on the context in which they are applied. It becomes then very difficult to choose which particular method is most likely to provide the best results for a given simulation.

To address this issue in SOFA we have introduced, for the Behavior Model, a finer level of granularity than what is described in section 2.1. This permits for instance to switch from one solver to another in order to see the change in performance or robustness of the simulation, or to test different constitutive models. These changes can be done in a matter of seconds, without having to recompile any of the code, by simply editing an XML file. To achieve this level of flexibility, we have defined a series of generic primitives, or *components*, that are common to most physics-based simulations: `DoF`, `Mass`, `Force Field`, and `Solver`.

The `DoF` component describes the degrees of freedom, and their derivatives, of the object. This includes positions, velocities, accelerations, as well as other auxiliary vectors. The `Mass` component represents the mass of the object. Depending on the model, the mass can be represented by a single value – all the DoFs have the same mass, a vector – the DoFs have a different mass, or even a matrix as used in complex finite element models. The `Force Field` describes both internal forces associated with the constitutive equations of the model, and external forces that can be applied to this object. A variety of forces are currently derived from the abstract Force Field representation, including springs, linear and co-rotationnal FEM [5,6], Mass-Tensor, and Smoothed Particle Hydrodynamics (SPH). The `Solver` component handles the time step integration, i.e. advancing the state of the system from time t to time $t + \Delta t$. To this end, the solver sends requests to the other components to execute operations such as summation of forces, computation of accelerations, and vector operations on the DoFs such as $x = x + v \cdot \Delta t$. Currently SOFA integrates explicit Euler and Runge-Kutta 4 solvers, as well as implicit conjugate-gradient based Euler solver [2].

2.3. Scene Graph Representation

Building and maintaining the relations between all the elements of a simulation can become quite complex. Reusing concepts from the graphics community, we decided for a homogeneous scene-graph representation, where each component is attached to a node of a tree structure. While components are user-defined and can be extended at will, internal nodes are all the same. They only store pointers to their local components, as well as their parent and children nodes. This simple structure enables to easily visit all or a subset of the components in a scene, and dependencies between components are handled by retrieving sibling components attached to the same node. For instance, a `Force Field` component can access the `DoF` component by getting its pointer from the node. The scene-graph can also be dynamically reorganized, allowing for instance the creation of groups of interacting objects. Such groups can then be processed as a unique system of equations by the solver, thus permitting to efficiently handle stiff contact forces. Another advantage of using a scene-graph is that most computations performed in the simulation loop can be expressed as a traversal of the scene-graph. This traversal is called an *action* in SOFA. For instance, at each time step, the simulation state is updated by sending an `Animate` action to all `Solver` components. Each `Solver` then forwards requests to the appropriate components by recursively sending actions within its sub-tree.

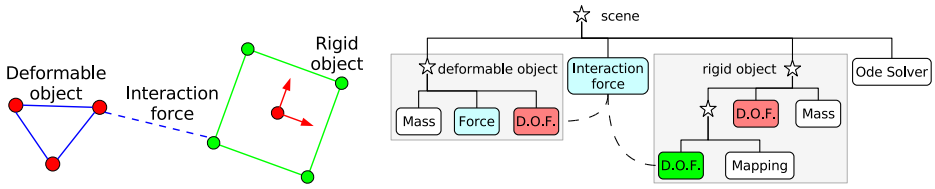

Figure 2. Left: two interacting bodies. The DoFs are shown as circles, and the forces as lines. A solid line describes an internal force, a dotted line an external force. Right: graph associated to the scene on the left. The nodes of the scene-graph, shown as stars, allow to model structured groups of components.

To illustrate the modularity in SOFA and the use of a scene-graph, we consider the example illustrated in Figure 2. In this example, two simulated objects – a rigid square and a simple Mass-Spring model – move through space and eventually collide. To compute the motion and deformation of the objects, we need to define for each of them a set of DoFs and a set of internal and external forces. The `DoF` component of the mass-spring model corresponds to the mass-points, while for the rigid object it corresponds to the position and orientation of the center of mass. This implies different data types for the DoFs of each object – a set of 3D vectors for the mass-spring and a 3D vector with a quaternion for the rigid object. Contacts between objects are possible through Collision Models associated with each object. The Collision Model for the mass-spring object consists of a set of vertices coincident with the DoFs of the object. The Collision Model for the rigid object – the square shape in Figure 2 – is rigidly attached to the body reference frame through a Mapping. The `Mapping` component is responsible for propagating the motion of the rigid body to the vertices of the Collision Model, and when collision occurs, the contact forces applied to the Collision Model are propagated back to the DoFs of the rigid body object. Since the vertices of the Collision Model do not coincide with the DoFs of the rigid object, we attach them to a different node of the scene-graph. However, as their motion is totally defined by the rigid body, they

are not independent so this new node is created as a child of the rigid body node. The interaction force acts on the collision model vertices, independently of whether they are actual or mapped DoFs. At this point, actions can be propagated through the scene-graph to simulate both objects as a combined mechanical system.

3. Results

We present here several examples of simulations developed using SOFA. These examples illustrate the diversity and flexibility of the SOFA framework, in particular the ability to have objects with different behavior interact together. We also demonstrate some early results on the validation of algorithms used for simulating deformable structures.

Laparoscopic Simulation: the primary target for SOFA being Medical Simulation, we have developed an early prototype of a laparoscopic simulation system in which the liver and intestines are modeled as deformable models which can be manipulated using a laparoscopic instrument and can collide with the ribs, as illustrated in Figure 3. The modularity of the SOFA architecture allows us to easily experiment different constitutive models for the organs. In this example the liver is modeled as a co-rotational FEM and the intestines as a spring-based FFD grid. The separation between Visual, Collision, and Behavior models allows us to generate visually appealing simulations at interactive rates.

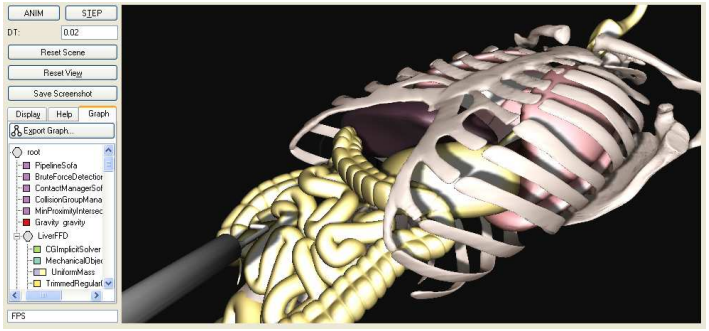

Figure 3. Simulation of laparoscopic surgery using SOFA at interactive rates (about 50Hz).

Quantitative validation and comparison of algorithms: comparing algorithms for soft-tissue deformation only makes sense if they are compared against reference models issued from the real world. To this end, we have built a cylinder using silicon gel of known material properties, and then applied controlled constraints to this object as it was being CT scanned. The resulting surface obtained after image processing is illustrated in Figure 4. This surface was used as a Visual Model to which various Behavior Models were assigned – mass-spring, co-rotational FEM, and linear FEM. It then becomes very easy to visually and quantitatively assess the accuracy of the various models.

Chain Links: handling interactions between heterogenous models is prone to stability issues. To test the robustness of different algorithms we experimented with falling chains where each link uses a different Behavior Model, as illustrated in Figure 5. No constraints between links were pre-defined, instead we relied on collision detection and stiff contact forces to handle the contacts. Using implicit integrator handling dynamically-created groups of interacting objects resulted in a stable simulation.

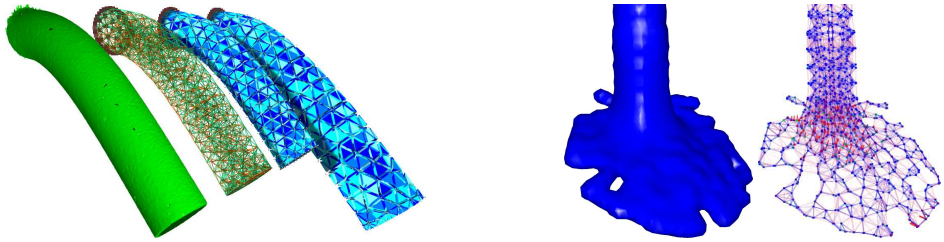

Figure 4. Left: surface of an actual soft cylindrical object compared to a mass-spring, co-rotational FEM, and linear FEM models, under the same constraints. Right: a fluid modeled in SOFA using a SPH method.

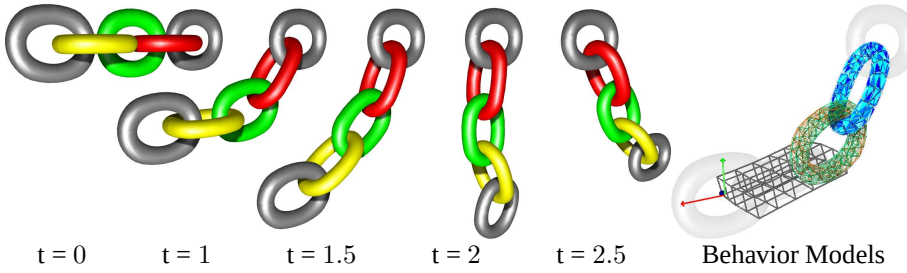

Figure 5. Animation of a chain combining a FEM model, a mass-spring model, a FFD grid, and a rigid body.

4. Conclusion and Future Work

The SOFA framework currently integrates, in the same environment, a variety of different algorithms, from springs and co-rotational FEM models to FFD deformation grids, as well as implicit and explicit solvers, and several collision detection methods, such as continuous or proximity-based algorithms. Our framework also supports hard constraints and stiff interaction forces, using implicit or multi-step explicit integrators that handle dynamically-created groups of interacting objects. Our future work includes the support for multi-processing, topological changes, and haptic feedback. The SOFA web site, *www.sofa-framework.org*, can be visited for more information on our most recent results.

Acknowledgments

We want to thank Sylvere Fonteneau, Damien Marchal, Xunlei Wu, Paul Neumann, Jeremie Dequidt, and Julien Lenoir for their contribution to the development of SOFA.

References

[1] J. Bacon, N. Tardella, J. Pratt, and J. English. The Surgical Simulation and Training Markup Language: An XML-Based Language for Medical Simulation. In *Proceedings of MMVR*, pages 37–42, 2006.

[2] D. Baraff and A. Witkin. Large steps in cloth simulation. In *Proceedings of SIGGRAPH*, 1998.

[3] T. Goktekin, M. Cenk Cavusoglu, and F. Tendick. Gipsi: An open source software development framework for surgical simulation. In *International Symposium on Medical Simulation*, pages 240–248, 2004.

[4] M. Kawasaki, M. Rissanen, N. Kume, Y. Kuroda, M. Nakao, T. Kuroda, and H. Yoshihara. VRASS (Virtual Reality Aided Simulation). In *www.kuhp.kyoto-u.ac.jp/ mi/research/vrass/index_en.shtml*.

[5] M. Muller and M. Gross. Interactive virtual materials. In *Graphics Interface'04*, pages 239–246, 2004.

[6] M. Nesme, Y. Payan, and F. Faure. Efficient, physically plausible finite elements. In *Eurographics*, 2005.

[7] K. Montgomery *et. al.* Spring: A general framework for collaborative, real-time surgical simulation. In *Proceedings of MMVR*, pages 23–26, 2002.