Modeling and Simulation of a Human Shoulder for Interactive Medical Applications

Miguel A. Otaduy Carlos Garre Jorge Gascón
Eder Miguel Álvaro G. Pérez Javier S. Zurdo
Universidad Rey Juan Carlos, Madrid

Abstract

Human joints, such as the shoulder, present intricate connections of anatomical elements such as bones, muscles, tendons, ligaments, and fat. The nature and arrangement of the various structures in the shoulder impose two main difficulties for interactive simulation: a large diversity of mechanical properties, ranging from hard bone to soft fat tissue, and complex contact situations. In this paper, we present a combination of representations, simulation methodology, and algorithms, which, altogether, provide the proper balance between simulation quality and performance for interactive medical applications. Unified representations for all dynamic objects and their dynamic state allow us to define coupling constraints and contact constraints in a general way. As a result, all dynamic objects can be simulated at once in a unified manner. We show the application of our algorithm to shoulder simulation in two medical settings: virtual arthroscopy and physiotherapy palpation.

1 Introduction

Virtual reality simulators provide medical practitioners with a non degradable, versatile, and realistic environment, in which novices may learn and try as much as desired. There is a vast amount of examples of medical simulators, as discussed in various survey papers [13, 12, 2].

But the true explosion of medical simulators is still to happen, and one of the major obstacles for this explosion is the difficulty to simulate in an interactive yet realistic manner the internal human anatomical structures. These anatomical structures present challenges such as a very diverse mechanical behavior, ranging from hard bone to soft fat tissue, and intricate contact situations. Contact, with its associated problems of collision detection, collision response, and friction handling, is often a computational bottleneck and, more importantly, a task with an unpredictable computational cost, which severely complicates interactive simulation.

In this paper, we present a combination of representations, simulation methodology, and algorithms, geared at producing efficient yet plausible simulation of intricate internal human anatomy. One of the key ingredients for our simulation methodology, described in Section 3, is the choice of appropriate representations for dynamics simulation, contact handling, and visualization. A generalized definition of dynamics representation allows us to handle the binding of surface representations, as well as the coupling of anatomical parts, all in an elegant and unified manner. In Section 4 we discuss a modeling pipeline to produce all the representations and the coupling. And the second key ingredient of our methodology, described in Section 5, is a contact handling algorithm that accounts in a unified yet efficient manner for diverse dynamic representations, their couplings, and contact constraints.

Human joints are one of the situations where the challenges of biomechanical simulation arise constantly, therefore we have selected the shoulder as the target example for demonstrating our results. Specifically, we show applica-

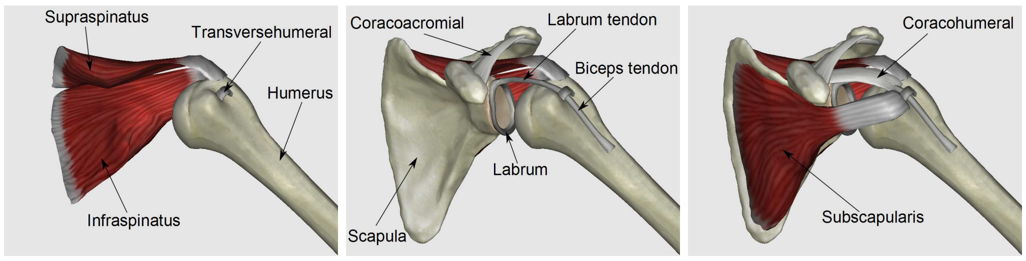

Figure 1: Layered display of the anatomical parts of the shoulder simulated in our examples.

tion of interactive shoulder simulation to virtual arthroscopy [3] and physiotherapy palpation [6], as depicted in Fig. 6. With our simulation methodology and carefully selected representations, we obtain interactive contact, deformations, and even haptic feedback, on intricate situations involving multi-way contact of several anatomical parts.

2 Related Work

Biomechanical modeling has seen a lot of success recently in computer graphics for the simulation of body parts such as the hand [24], the neck [11], the face [20], the torso [25, 5], or the complete upper body [10]. These models rely on highly detailed discretizations and geometrically accurate modeling of the anatomy, which imposes severe restrictions on their applicability to interactive simulation.

For interactive simulation of deformations, methods based on the linear co-rotational finite element formulation [15] are perhaps the ones that give a best balance between performance, robustness, and measurement-based parameterization. This last aspect is important when the behavior of a model should approximate the behavior of a real structure. Linear co-rotational FEM models can be accelerated by decoupling the simulation mesh from the visualization or collision detection mesh, using embedded meshes [14, 23, 16].

Another important aspect for a biomechanical simulation is contact handling. Two main approaches exist, penalty methods [1] and constraint-based methods [7, 18]. Even though they may have a higher computational cost, constraint-based methods provide higher robustness under stability issues. For the case of rigid bodies, they are used on several open-source libraries [4, 17]. It is also worth pointing out the SOFA open-source library [22], which, similar to our work, supports soft-tissue contact. One of the major issues that our work addresses is the efficient and easy coupling of objects with various mechanical properties. This has been addressed for coupling constraints by [21] using binding springs. We follow a similar approach and solve binding springs efficiently in a global conjugate gradient solve. Coupling through contact constraints is often addressed in the game engine field, with the runtime creation of contact islands [19].

3 Representations

We use separate representations for the three main tasks in the simulation, namely, dynamics, collisions, and visualization, which allows us to treat each task efficiently. The dynamics representation acts as the link between all three representations, and it completely defines the state of the collision and visualization representations. In this section, we describe the three representations, as well as the generic data structures and mathematics for binding them together.

3.1 Dynamics, Collisions, Visualization

In our algorithm, we define a *contact object* as the elementary dynamic object with distinct mechanical properties (e.g., a bone, a ligament, etc.). Its dynamics representation consists of a state vector $\mathbf{q}$ and a velocity vector $\mathbf{v}$ that fully define the dynamics of an object. In the general case, the velocity and state vectors are related as $\dot{\mathbf{q}} = \mathbf{G}\mathbf{v}$. All contact objects share a common interface from the software engineering point-of-view, which allows us to handle them all in a unified manner in the simulation algorithm to be described in Section 5. For rigid bodies, the state vector $\mathbf{q}$ consists of the position of the center of mass and a quaternion for the orientation, while the velocity vector $\mathbf{v}$ consists of the linear and angular velocities of the body. For deformable bodies, we use a linear co-rotational finite element formulation [15], with objects discretized using tetrahedral meshes. Then, the state vector is formed by the positions of mesh nodes, and the velocity vector is formed by the velocities of the nodes.

For collisions and visualization, we represent each contact object using a collection of triangle meshes. The topology of these meshes is fixed, and their geometry is fully defined by the positions of their vertices, which in turn are defined by the dynamics representation as we describe next.

3.2 Binding Dynamics and Surfaces

The position of every vertex in a triangle mesh is computed using a generic *point* entity. In essence, a point binds a vertex and a generic contact object, by a definition of the vertex position as $\mathbf{p} = f(\mathbf{q})$. For rigid bodies, the vertex position is defined as $\mathbf{p} = \mathbf{c} + \mathbf{R}\mathbf{r}$, where $\mathbf{c}$ and $\mathbf{R}$ are the position and orientation of the rigid body, and $\mathbf{r}$ is the position of the vertex in the body's local reference system. For deformable bodies, we assume each surface vertex to be embedded inside a tetrahedron. Then, the vertex position is defined as $\mathbf{p} = \sum_{i=1}^{4} w_i \mathbf{q}_i$, where the 4 $\mathbf{q}_i$ values are the positions of tetrahedral nodes, and the w_i are barycentric coordinates. Full details about the construction and embedding of the tetrahedral mesh are given in the next section.

The point entity also relates the velocities of mesh vertices and the velocity vector of a contact object, by simple differentiation of the position, $\dot{\mathbf{p}} = \mathbf{J}\mathbf{v}$, with $\mathbf{J} = \frac{\partial \mathbf{p}}{\partial \mathbf{q}}\mathbf{G}$. Each specific point entity, depending on the type of contact object it acts on, stores this relationship in a compact manner. For deformable objects, for example, $\mathbf{J}$ is a sparse matrix where the only non-zero columns are those due to the tetrahedral nodes that affect the surface vertex. It is important to point out that velocities are always linearly related.

But one of the main features of our generic point entity definition is that it allows us to define forces and constraints on surface vertices in a unified manner, independently of the type of contact object. Given the relationship between the velocity vector and the velocity of a vertex, $\mathbf{J}$, forces on a contact object can be computed from surface forces as $\mathbf{F}_{\mathbf{q}} = \mathbf{J}^T \mathbf{F}_{\mathbf{p}}$. The linear relationship between velocities and between forces is the classic *manipulator Jacobian* from robotics.

For visualization purposes, we store mesh connectivity in the GPU by exploiting buffer objects. Every time a contact object is modified, we simply need to send the state vector $\mathbf{q}$ to the GPU.

4 Modeling

In this section, we list the anatomical parts of the shoulder that we have accounted for in our simulations, and we describe our modeling pipeline to produce the simulation scenario where the various representations of the different parts are integrated.

4.1 Description of Anatomical Parts

The anatomy of the shoulder is full of diverse parts such as bones, muscles, ligaments, cartilages or tendons. The result is a compact and heterogeneous volume where it is not easy for unexperienced persons to discern one part from another. In order to develop useful and interesting applications, we need to simplify the anatomical complexity and focus on those parts that are meaningful for our application.

For example, in the case of arthroscopy, there are some bulky parts that are not inter-

Id	Anatomical part	Render	CD	FEM	Couplings
1	Scapula	10224	105	NA	3, 4, 5, 6(x2), 7, 9
2	Humerus	5189	112	NA	2(x2), 3, 4, 5, 7, 8(x2), 11
3	Supraspinatus	1716	86	87	1, 2
4	Infraspinatus	2136	96	129	1, 2
5	Subscapularis	2424	100	158	1, 2
6	Coracoacromial	1632	146	65	1(x2)
7	Coracohumeral	522	60	86	1, 2
8	Transversehumeral	264	28	NA	2(x2)
9	Labrum	636	120	96	1, 10
10	Labrum Tendon	128	128	122	9, 11
11	Biceps Tendon	264	264	NA	2, 10

Table 1: List of anatomical parts simulated in the arthroscopy example, with the number of triangles of their visualization surface (Render), number of triangles of the collision surface (CD), number of tetrahedra of the dynamic representation (FEM), and list of couplings to other parts (indicated by their ids). 'x2' means that the coupling with another anatomical part takes place at two locations. The number of tetrahedra does not apply for rigid bodies.

esting for the surgeon, such as the subacromial bursa or the deltoid muscle. In Fig. 1, we show the parts that we have accounted for in our arthroscopy example. We model the bones, i.e., the scapula and humerus, as rigid bodies. Moreover, due to their limited range of motion, we also model the biceps tendon and the transversehumeral as rigid bodies. All other parts are modeled as soft bodies. Table 1 indicates pairs of parts that are coupled using zero-length coupling springs. Note that some parts are coupled at two different locations to the same bone. When a large surface of a soft body is coupled to a bone, we disable collision detection between the two objects, since there is no relative motion between them. We do this, for example, in the coupling between the scapula and the labrum.

For physiotherapy palpation, on the other hand, the criterion for selecting the interesting parts is almost the opposite as for arthroscopy, because the practitioner focuses mainly on the outer anatomical layers. Therefore, for physiotherapy palpation we incorporate the deltoid muscle.

4.2 Model Creation Pipeline

The input to our pipeline is a set of triangle meshes that describe the surfaces of the various anatomical parts. These meshes can be obtained by scanning real parts with a standard 3D scanner, or by manual authoring. We use these meshes as the visualization representations in our examples.

For collision handling purposes, we apply standard mesh simplification techniques to obtain low-resolution approximations that can be efficiently handled interactively. Then, for the soft-tissue parts, we create the tetrahedral meshes that define the dynamic representation by embedding both the visualization and collision meshes. We start by enclosing the surface meshes with a bounding box, we subdivide it regularly to the desired cell resolution, decompose each cubic cell into five tetrahedra, and finally we eliminate those tetrahedra that do not intersect the volume enclosed by the visualization and collision meshes. Thanks to the embedding-based simulation, plausible deformations are possible even with rather coarse tetrahedral meshes. Fig. 2 shows the visualization, collision, and dynamics meshes for the coracoacromial ligament. The resolution of all meshes for the arthroscopy example is listed in Table 1.

In order to define couplings between a soft body and a rigid bone, we manually select tetrahedral nodes of the soft body that should

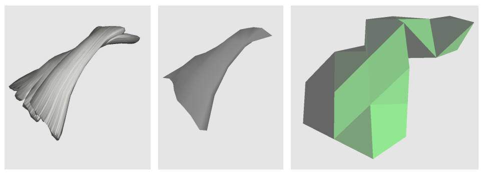

Figure 2: From left to right, visualization, collision, and dynamic meshes for the coracoacromial ligament.

be coupled to the bone, and we set zero-length binding springs at those nodes. Both end-points of a binding spring are fully defined by the state of their corresponding contact objects, using the point entity defined in Section 3.2. For couplings between soft bodies, we manually select tetrahedral nodes from both bodies, and set binding springs at those locations. Nodes from two bodies a and b are typically not collocated, therefore, one end-point of the binding spring is defined directly by the state of a tetrahedral node, while the other end-point is defined through barycentric interpolation inside the enclosing tetrahedron in the other body.

Due to the intricate layout of anatomical parts, it would be a daunting modeling task to ensure that all parts are intersection-free at their undeformed state. Instead of enforcing this, we allow the objects to intersect in the undeformed state, but we define an initialization state where they are intersection-free. We do this by deforming each tetrahedral mesh (and thus the visualization and collision meshes as well) using a cage-based deformation technique [9], as depicted in Fig. 3. In the simulation, the tetrahedral meshes are then initialized at a deformed state, and as soon as the simulation starts they move to a minimum energy situation.

5 Simulation Algorithm

In this section we explain the main features of the algorithm that allows the simulation, in a unified yet efficient manner, of complex anatomical scenarios composed of objects with diverse mechanical behavior. At the same time, this algorithm handles elegantly coupling and contact constraints, making them independent of the objects that they act on.

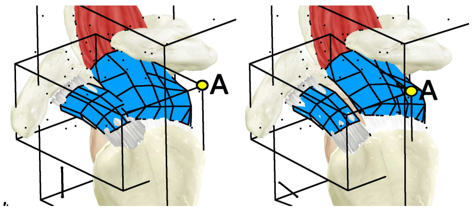

Figure 3: Cage-based deformation of the coracohumeral ligament to ensure a collision-free initial state. In blue, the collision meshes. Notice how in the undeformed state (on the left), the collision meshes are intersecting.

5.1 Implicit Integration of Dynamics

Given state and velocity vectors $\mathbf{q}$ and $\mathbf{v}$ that group the state and velocity of all contact objects in the scene, the dynamics of the simulation are discretized with the ODEs:

$$\begin{aligned} \mathbf{M}\dot{\mathbf{v}} &= \mathbf{F}, \\ \dot{\mathbf{q}} &= \mathbf{G}\mathbf{v}, \end{aligned} \tag{1}$$

where $\mathbf{M}$ denotes the mass matrix and $\mathbf{F}$ is the force vector. We numerically integrate the ODEs using the (implicit) backward Euler method with linear approximation of forces, which yields a velocity update

$$\begin{aligned} &\mathbf{A}\mathbf{v} = \mathbf{b}, \qquad \text{with} \\ &\mathbf{A} = \mathbf{M} - \Delta t \frac{\partial \mathbf{F}}{\partial \mathbf{v}} - \Delta t^2 \mathbf{G} \frac{\partial \mathbf{F}}{\partial \mathbf{q}}, \\ &\mathbf{b} = \Delta t \mathbf{F}(\mathbf{v_0}, \mathbf{q_0}) + \left(\mathbf{M} - \Delta t \frac{\partial \mathbf{F}}{\partial \mathbf{v}} \right) \mathbf{v_0}. \end{aligned} \tag{2}$$

Eq. (2) is solved using a Conjugate Gradient (CG) solver, and the result is the unconstrained velocity of the contact objects.

In a simulation with three independent objects, Eq. (2) can be writen as:

$$\begin{pmatrix} \mathbf{A}_{11} & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{A}_{22} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbf{A}_{33} \end{pmatrix} \begin{pmatrix} \mathbf{v}_1 \\ \mathbf{v}_2 \\ \mathbf{v}_3 \end{pmatrix} = \begin{pmatrix} \mathbf{b}_1 \\ \mathbf{b}_2 \\ \mathbf{b}_3 \end{pmatrix}. \tag{3}$$

All non-diagonal terms of $\mathbf{A}$ are zero because there are no couplings (i.e. no forces) acting between different objects in the simulation. Our algorithm makes use of this fact and uses the CG solver efficiently, solving Eq. (2) for each object independently.

5.2 Coupling Islands

For each pair of anatomical structures that are solidly attached, we define a *coupling* between their corresponding contact objects. Our definition of coupling is general and is able to handle any pair of contact objects. Specifically, each coupling consists of two general *semicouplings* and, from a software engineering perspective, we define different semicoupling implementations based on the types of contact objects in our simulation.

For each coupling, we set zero-length springs between the two coupled contact objects, as shown in Fig. 4. These springs add new forces to the system, modifying the structure of the terms in Eq. (2). With coupled objects as in Fig. 4, the new system structure is:

$$\begin{pmatrix} \mathbf{A}_{11} & \mathbf{A}_{12} & \mathbf{0} \\ \mathbf{A}_{21} & \mathbf{A}_{22} & \mathbf{A}_{23} \\ \mathbf{0} & \mathbf{A}_{32} & \mathbf{A}_{33} \end{pmatrix} \begin{pmatrix} \mathbf{v}_1 \\ \mathbf{v}_2 \\ \mathbf{v}_3 \end{pmatrix} = \begin{pmatrix} \mathbf{b}_1 \\ \mathbf{b}_2 \\ \mathbf{b}_3 \end{pmatrix}. \quad (4)$$

The system is no longer block-diagonal, and the right-hand side $\mathbf{b}$ is also modified. Given a zero-length spring between two points $\mathbf{p}_a$ and $\mathbf{p}_b$, the spring force acting on point $\mathbf{p}_a$ is $\mathbf{F}_{\mathbf{p}_a} = -k(\mathbf{p}_a - \mathbf{p}_b)$, and the force acting on object a is $\mathbf{F}_{\mathbf{q}_a} = \mathbf{J}_a^T \mathbf{F}_{\mathbf{p}_a}$. Off-diagonal terms in $\mathbf{A}$ are due to non-zero derivatives of the form $\frac{\partial \mathbf{F}_{\mathbf{q}_a}}{\partial \mathbf{q}_b} = k\mathbf{J}_a^T \frac{\partial \mathbf{F}_{\mathbf{p}_b}}{\partial \mathbf{q}_b}$. These derivatives can be efficiently computed in a unified manner for arbitrary couplings making use of our point entities defined in Section 3.2.

The solution to the velocity update can no longer be executed by doing an independent CG solve for each object. A naïve approach would then compute one global CG solve for the complete system, but we optimize this by identifying sets of objects that are coupled to form a *coupling island.* Two contact objects a and b belong to the same coupling island if and only if they share at least one coupling. Then, an independent CG solve can be executed for each coupling island.

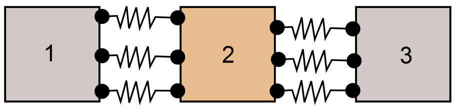

Figure 4: Example of coupled objects using springs. Objects (1) and (3) represent rigid bodies, while object (2) represents a deformable body.

Assuming that couplings are not dynamically created or eliminated, we define coupling islands as a preprocess. In the modeling stage described in Section 4, we first define the vector of contact objects, and then a vector of coupling entities. Once all structures and interactions are defined, we initialize coupling islands with individual contact objects, and we grow these coupling islands by traversing the vector of coupling entities.

At runtime, we need to assemble the system $(\mathbf{A}, \mathbf{b})$ of each coupling island prior to the CG solve. In order to do this, we first assemble the system matrices and right-hand-sides of the individual contact objects and coupling entities, and then we merge them.

5.3 Contact Islands

In order to handle contact efficiently yet robustly, we follow the constraint-based formulation in [18]. Given a pair of contact points $\mathbf{p}_a$ and $\mathbf{p}_b$, we define a non-penetration constraint as an algebraic inequality $g(\mathbf{p_a}, \mathbf{p_b}) = \mathbf{n}^T(\mathbf{p_a} - \mathbf{p_b}) \geq 0$, where $\mathbf{n}$ is the contact normal.

Constraints are then formulated semi-implicitly and transformed into velocity constraints like $\mathbf{J_a v_a} + \mathbf{J_b v_b} \geq \mathbf{c}$, where the Jacobians are computed using the general point entities described in section Section 3.2. These velocity constraints, together with Signorini's contact condition [7], are added to Eq. (2), leading to a Mixed Linear Complementarity Problem (MLCP), where contact forces are expressed as $\mathbf{J}^T\lambda$. The full MLCP that defines

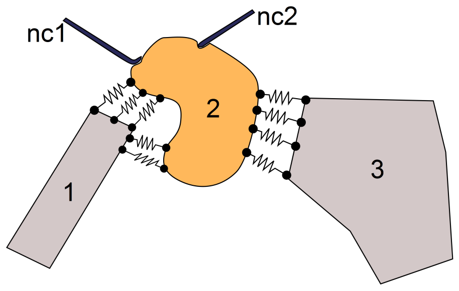

Figure 5: A contact island composed of two individual contact objects, i.e., the tools ($nc1$) and ($nc2$), and one coupling island, formed by contact objects (1), (2) and (3).

the constrained velocities can be expressed as:

$$\begin{aligned} &\mathbf{A}\mathbf{v} = \mathbf{J}^T\mathbf{v} + \mathbf{b}, \\ &\mathbf{0} \leq \lambda \perp \mathbf{J}\mathbf{v} \geq \mathbf{c}. \end{aligned} \tag{5}$$

We solve this MLCP using an iterative scheme composed of two nested loops, as explained in [18]. First, a Jacobi iteration over the velocities defines the outer loop and decomposes matrix $\mathbf{A}$ into its diagonal and lower and upper triangular parts, $\mathbf{A} = \mathbf{D_A} - \mathbf{L_A} - \mathbf{U_A}$. With this decomposition, the iterative MLCP is transformed into its corresponding LCP:

$$\begin{aligned} &\mathbf{0} \leq \lambda \perp \mathbf{B}\lambda \geq \mathbf{d}, \qquad \text{with} \\ &\mathbf{B} = \mathbf{J}\mathbf{D_A}^{-1}\mathbf{J}^T, \\ &\mathbf{d} = \mathbf{c} - \mathbf{J}\mathbf{D_A}^{-1}\left(\mathbf{b} + \left(\mathbf{L_A} + \mathbf{U_A}\right)\mathbf{v}\right). \end{aligned} \tag{6}$$

Then, the LCP is solved to compute λ using the Projected Gauss-Seidel (PGS) method (which represents the inner loop), and the current iteration of the constrained velocity is obtained.

Instead of solving one large MLCP for the complete scene, we identify *contact islands* and formulate and solve one MLCP for each contact island. Two coupling islands belong to the same contact island if and only if they share at least one contact constraint. For the example in Fig. 5, where a single contact island is composed of two individual contact objects and one coupling island, the system matrix can be written as

$$\mathbf{A} = \begin{pmatrix} \mathbf{A_c} & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{A_{nc1}} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbf{A_{nc2}} \end{pmatrix},$$

with the submatrix for the coupling island expressed as

$$\mathbf{A_c} = \begin{pmatrix} \mathbf{A_{11}} & \mathbf{A_{12}} & \mathbf{0} \\ \mathbf{A_{21}} & \mathbf{A_{22}} & \mathbf{A_{23}} \\ \mathbf{0} & \mathbf{A_{32}} & \mathbf{A_{33}} \end{pmatrix}.$$

It is important to note that the off-diagonal terms of matrix $\mathbf{A}$ do not have much impact on the efficiency of the constrained solve, as opposed to the unconstrained solve discussed earlier. The reason is that the lower and upper triangular parts of $\mathbf{A}$ affect only the right-hand side of the LCP, $\mathbf{d}$, as shown in Eq. (6), and are not visited during the iterations of PGS.

In contrast to the unconstrained update, the system matrix for each contact island is not assembled explicitly. Instead, an object-oriented processing is followed, accessing each object's data when required by the iterative solver. Based on an implementation of contact islands that deals with contact objects directly, we have extended it to seamlessly deal with coupling islands. From a software engineering perspective, we achieve this through abstraction, by ensuring that coupling islands share the same interface as contact objects.

6 Results

We have executed our experiments on a quad-core 2.4 GHz PC with 3 GB of memory (although we have only used two cores, for the visual and haptic loops) and a GeForce 8800 GTS. For haptic rendering, we have used the method by [8]. All the modeling tasks and the cage-based deformation have been executed using Blender 3D, while the real-time renderings have been performed with OGRE.

Fig. 6 shows images of the two applications where we have tested our interactive shoulder simulator, arthroscopy and physiotherapy palpation. Our arthroscopy example does not include a realistic portal-based interaction [3],

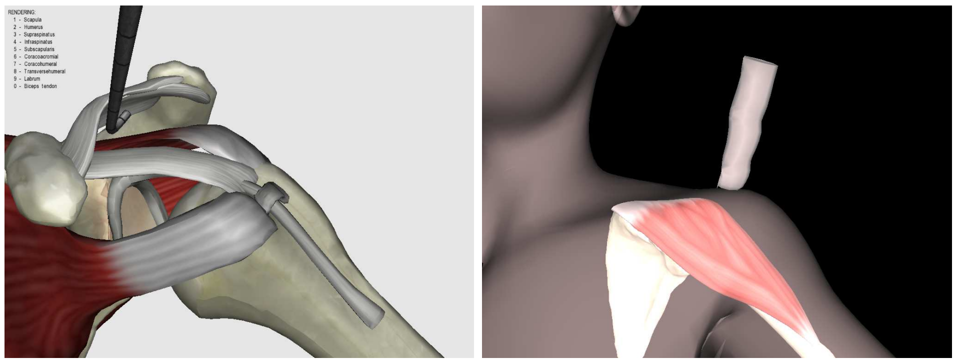

Figure 6: Interactive simulations in virtual arthroscopy (left) and physiotherapy palpation (right).

but this was not the purpose of our work. In the palpation application, we simulate two finger models, one touching external geometry and another one the internal anatomy. The finger models are linked through a spring that models skin stiffness. Please watch the accompanying video for dynamic sequences of intensive contact interactions, as well as a tutorial of the modeling pipeline.

In the arthroscopy example, the scene has 50 contacts at rest-state. During some sample haptic interactions that we performed, the average number of contacts was 65, and it reached a maximum of 153. The complete simulation runs at an average of 50 fps. Note that we ensure a 1 kHz haptic update rate thanks to a multi-rate haptic rendering approach.

7 Discussion and Future Work

The main conclusion that can be extracted from our results is that our simulation methodology is successful in achieving interactive simulation of complex human joints for medical applications. Key to this success are the optimization of representations, and the efficient handling of couplings and contacts in the constraint-based dynamics solver.

The approximations that we carry out have some implications on the accuracy of the simulation. For example, the couplings between the various parts are not fully anatomically correct, and sometimes we partially eliminate the possibility of muscles to slide on top of bones. In order to fully simulate the shoulder anatomy, we would need to incorporate the bursa, but this structure produces even more intensive contact situations that would be difficult to handle interactively. The fact that our contact handling is based on iterative solvers prevents us from guaranteeing a certain minimum frame rate, although we did not see this to be a problem in practice.

Besides addressing the limitations of our approximations, there are many possible avenues for future work. We are working together with both arthroscopy and physiotherapy experts to assess the quality of our models and guide further developments. In arthroscopy, many medically interesting interactions are related to tissue cutting and to suture. Therefore, a fully fleshed arthroscopy simulation would require interactive simulation of topological changes and contact with thread models.

Acknowledgments

We would like to thank the anonymous reviewers for their helpful comments, Gonzalo Mora and Beatriz Tierno for clinical advice, the GMRV team at URJC and the Healthcare team at GMV for discussions. This work has been funded in part by the Spanish Dept. of Science and Innovation (projects TIN2009-07942 and PSE-300000-2009-5).

References

[1] J. Barbič and D. James. Time-critical distributed contact for 6-DoF haptic rendering of adaptively sampled reduced deformable models. In *2007 ACM SIGGRAPH / Eurographics Symposium on Computer Animation*, pages 171–180, Aug. 2007. 2

[2] C. Basdogan, M. Sedef, M. Harders, and S. Wesarg. Vr-based simulators for training in minimally invasive surgery. *IEEE Comput. Graph. Appl.*, 27(2):54–66, 2007. 1

[3] S. Bayona, M. García, C. Mendoza, and J. Fernández-Arroyo. Shoulder arthroscopy training system with force feedback. *Proceedings of Medical Information Visualisation*, pages 71–76, 2006. 1, 6

[4] BULLET. Bullet Physics Library. `http://bulletphysics.org/`. 2

[5] P. C. DiLorenzo, V. B. Zordan, and B. L. Sanders. Laughing out loud: Control for modeling anatomically inspired laughter using audio. *ACM Trans. Graph.*, 27(5):125:1–125:8, Dec. 2008. 2

[6] M. Dinsmore, N. Langrana, and G. Burdea. Issues related to real-time simulation of a virtual knee joint palpation. *Proceedings of Virtual Reality and Medicine, The Cutting Edge*, pages 16–20, 1994. 1

[7] C. Duriez, F. Dubois, A. Kheddar, and C. Andriot. Realistic haptic rendering of interacting deformable objects in virtual environments. *Proc. of IEEE TVCG*, 12(1), 2006. 2, 5.3

[8] C. Garre and M. A. Otaduy. Haptic rendering of complex deformations through handle-space force linearization. In *Proc. of World Haptics Conference*, mar 2009. 6

[9] P. Joshi, M. Meyer, T. DeRose, B. Green, and T. Sanocki. Harmonic coordinates for character articulation. *ACM Transactions on Graphics*, 26(3):71:1–71:9, July 2007. 4.2

[10] S.-H. Lee, E. Sifakis, and D. Terzopoulos. Comprehensive biomechanical modeling and simulation of the upper body. *ACM Trans. Graph.*, 28(4):99:1–99:17, Aug. 2009. 2

[11] S.-H. Lee and D. Terzopoulos. Heads up!: biomechanical modeling and neuromuscular control of the neck. *ACM Trans. Graph*, 25(3):1188–1198, July 2006. 2

[12] P. Leskovsky, M. Harders, and G. Szekely. A web-based repository of surgical simulator projects. *Stud Health Technol Inform*, 119, 2006. 1

[13] A. Liu, F. Tendick, K. Cleary, and C. Kaufmann. A survey of surgical simulation: applications, technology, and education. *Presence: Teleoper. Virtual Environ.*, 12(6):599–614, 2003. 1

[14] N. Molino, R. Bridson, J. Teran, and R. Fedkiw. A crystalline, red green strategy for meshing highly deformable objects with tetrahedra. *International Meshing Roundtable*, pages 103–114, 2003. 2

[15] M. Müller and M. Gross. Interactive virtual materials. *Proc. of Graphics Interface*, 2004. 2, 3.1

[16] M. Nesme, P. G. Kry, L. Jeřábková, and F. Faure. Preserving topology and elasticity for embedded deformable models. *ACM Trans. Graph.*, 28(3):52:1–52:9, July 2009. 2

[17] ODE. Open Dynamics Engine. `http://www.ode.org/`. 2

[18] M. A. Otaduy, R. Tamstorf, D. Steinemann, and M. Gross. Implicit contact handling for deformable objects. *Computer Graphics Forum*, 28(2):559–568, Apr. 2009. 2, 5.3, 5.3

[19] E. G. Parker and J. F. O'Brien. Real-time deformation and fracture in a game environment. In *2009 ACM SIGGRAPH / Eurographics Symposium on Computer Animation*, pages 156–166, 2009. 2

[20] E. Sifakis, I. Neverov, and R. Fedkiw. Automatic determination of facial muscle activations from sparse motion capture marker data. *ACM Trans. Graph*, 24(3):417–425, Aug. 2005. 2

[21] E. Sifakis, T. Shinar, G. Irving, and R. Fedkiw. Hybrid simulation of deformable solids. In *2007 ACM SIGGRAPH / Eurographics Symposium on Computer Animation*, pages 81–90, Aug. 2007. 2

[22] SOFA. Simulation Open Framework Architecture. `http://www.sofa-framework.org/`. 2

[23] J. Spillmann, M. Wagner, and M. Teschner. Robust tetrahedral meshing of triangle soups. *Proc. Vision, Modeling, Visualization*, pages 9–16, 2006. 2

[24] S. Sueda, A. Kaufman, and D. K. Pai. Musculotendon simulation for hand animation. *ACM Trans. Graph.*, 27(3), Aug. 2008. 2

[25] J. Teran, E. Sifakis, S. S. Blemker, V. Ng-Thow-Hing, C. Lau, and R. Fedkiw. Creating and simulating skeletal muscle from the visible human data set. *IEEE Trans. on Visualization and Computer Graphics*, 11(3):317–328, May/June 2005. 2