Soft Hand Simulation for Smooth and Robust Natural Interaction

http://mslab.es/clap

Mickeal Verschoor*
Universidad Rey Juan Carlos

Daniel Lobo†
Universidad Rey Juan Carlos

Miguel A. Otaduy‡
Universidad Rey Juan Carlos

Abstract

Natural hand-based interaction should feature hand motion that adapts smoothly to the tracked user's motion, reacts robustly to contact with objects in a virtual environment, and enables dexterous manipulation of these objects. In our work, we enable all these properties thanks to an efficient soft hand simulation model. This model integrates an articulated skeleton, nonlinear soft tissue and frictional contact, to provide the realism necessary for natural interaction. Robust and smooth interaction is made possible by simulating in a single energy minimization framework all the mechanical energy exchanges among elements of the hand: coupling between the hand's skeleton and the user's motion, constraints at skeletal joints, nonlinear soft skin deformation, coupling between the hand's skeleton and the soft skin, frictional contact between the skin and virtual objects, and coupling between a grasped object and other virtual objects. We have put our effort on describing all elements of the hand that provide for realism and natural interaction, while ensuring minimal and bounded computational cost, which is key for smooth and robust interaction. As a result, we accomplish hand simulation as an asset that can be connected to diverse input tracking devices, and seamlessly integrated in game engines for fast deployment in VR applications.

Keywords: Hand Interaction, VR, Simulation

Index Terms: Computing methodologies—Computer graphics—Graphics systems and interfaces—Virtual reality; Human-centered computing—Human computer interaction (HCI)—Interaction techniques

1 Introduction

Hands are our principal means for interacting with the world. Due to their importance, it is no surprise that the scientific community has witnessed many methods for tracking [28], simulation [37], animation [23], robotics [4], or touch [9] based on the hand. All these areas could benefit from a smooth, robust and efficient hand simulation model, which could be integrated in particular applications.

For hand-based interaction with virtual objects, hand tracking is complemented with some interaction method that maps the unconstrained pose of the user's real hand to a pose of the virtual hand that is constrained by virtual objects. Two major families of interaction methods are currently used, gesture-based or physics-based, but neither of them has yet succeeded to achieve natural interaction. Gesture-based interaction methods lack accuracy, and while sufficient for the animation of grasp-and-release actions, they fail to support dexterous manipulation with hand poses that change smoothly during grasp or exploration actions. Physics-based interaction methods, on the other hand, lack robustness. They let the physics of frictional contact dictate the interaction between the hand and virtual objects, which can potentially produce very natural motions. However, due to high computational complexity, they must trade accuracy for interactivity. Methods that aim for high accuracy are not robust, while methods that favor interactivity fail to reproduce fine interactions. Our work can be positioned in between, where we aim for interactivity while keeping the method reasonably accurate for the performed tasks. Therefore, we opt for methods that approximate the true behavior of skin deformation and friction.

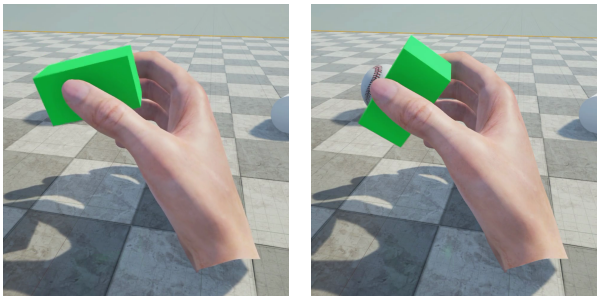


Figure 1: Natural and intuitive real-time hand-based interaction. Fine dexterous manipulation is possible thanks to the efficient and robust simulation of a soft hand model.

In our work, we have designed a physics-based hand simulation model that includes all the important elements for smooth and natural interaction, and it does so using computationally efficient methods for robust interaction. We design an energy-based formulation for all the relevant mechanical and interaction elements of the hand, and solve hand dynamics using a single energy-minimization formulation, see Sect. 3. The simulated hand model includes the following key elements: an articulated skeleton, nonlinear soft tissue surrounding the skeleton, skeleton-and-tissue coupling, frictional contact with the virtual environment, and coupling to the tracker input.

By simulating the hand skeleton, instead of directly applying the tracker input to the skeleton, our method absorbs the difference between the unconstrained tracked pose and the virtual hand pose constrained by virtual objects. We refer to our method as *elastic tracking*. It succeeds to limit the tracking forces acting on the hand's tissue, and thus results in natural hand motion and deformation.

Unlike most methods, which simulate articulated skeletons using constraint-based methods, we model joints using stiff energies. This choice could introduce some constraint error, but it is imperceptible in practice, and favors the application of the efficient energy-minimization solver.

Furthermore, our nonlinear tissue model produces a soft tissue behavior under low contact forces, together with incompressibility under high contact forces. This nonlinear behavior improves the quality of the deformed hand, i.e., the tissue is not subject to local collapse of the underlying mesh, resulting in a robust simulation and a smooth interaction. We simulate the tissue using a linear co-rotational FEM method, augmented with deformation limits that add a stiff penalty term. Our nonlinear elasticity model requires only minor changes to the regular linear co-rotational method.

*email: mickeal.verschoor@urjc.es
†email: wolfyllow@gmail.com
‡email: miguel.otaduy@urjc.es

A major result of our method is its ability to simplify the integration with off-the-shelf physics engines, which allows users to easily interact with virtual worlds through their hands. To achieve this, two instances of the object are simulated, one within the energy-minimization solver, and another one in the physics engine. We implement a coupling between the two instances of the object, which allows hand forces to be transmitted to the virtual environment, and other environment contacts to be transmitted to the hand.

By leveraging the easy integration with physics engines and input trackers, we have implemented our soft hand simulation model as a regular asset of a game engine. We summarize in Sect. 4 the implementation of CLAP, a library that includes the hand physics solver together with interface modules. As a result, we obtain a hand asset that can be seamlessly integrated in VR applications developed within a game engine.

As shown in Fig. 1, our soft hand model enables natural and intuitive real-time hand-based interaction. Please watch the accompanying video, where we demonstrate many robust, dexterous manipulation examples.

2 Related Work

The simulation of hands has received attention from multiple angles. In robotics, several works have developed methods for grasping simulation and control, including multi-purpose libraries [21, 26, 41]. However, in this case the focus is not on the hand's anthropomorphic accuracy. Throughout the years the simulation of humans and the human skin made a large development towards realism. The first models of human hands were solely based on rigid bodies and provided some form of force feedback to the user and a coupling between an articulated and tracked hand was made [3, 16, 29]. Although grasping of objects was possible, the absence of a deformable skin did not allow for accurate simulation of frictional constraints.

Simulating a human hand that involves a deformable skin requires one to simulate all different aspects that ultimately form the human hand. This includes the skeleton and the skin. In those simulations, it is important that each different aspect is coupled properly with all other parts such that a realistic behavior is obtained, which makes the problem very challenging to solve. Depending on the required level of detail, various approaches exist. Highly detailed approaches are for example found in [37], where the hand is simulated in a very high level of detail, which includes for example the muscles. Other approaches perform the deformation in a geometric way, but deformations due to physical interactions are often not modeled [18, 20]. Other methods use data-driven approaches for modeling the deformation of the hand by learning from examples. Since these methods are not completely physics-based, it is difficult to obtain a good deformation while the hand is in contact with an object, [19, 22, 33].

Using detailed hand simulations with deformable skin in interactive and real-time applications requires a trade-off between realism and computational costs. To reduce the computational complexity of simulating deformable hands, often only the fingers were simulated. This greatly reduces the complexity of the used model [2, 5, 8, 42]. Realistic simulations of the human hand incorporate a simulation of the rigid skeleton with a deformable skin connected to the bone structure. Methods that connect a portion of the deformable skin to the skeleton are for example [7, 15, 35]. More detailed approaches also simulate parts of the deformation of the skin, like [34]. Some other methods limit complexity by handling contact between the hand and objects efficiently, e.g., by aggregating contact constraints [38].

A further improvement consists of coupling the simulation of the whole skin to the bones through two-way coupling. This means that forces in the skin will propagate to the bones, and the bones can generate, as a result, forces on the skin. Such approaches greatly improve the realism of deformable hands. The approach of [9] performs the coupling between the bones and the skin in two steps. First, an approximate configuration of the skeleton is computed, taking constraints on the joints into account. Next the configuration of the skin is computed given the configuration of the bones, and the bones are allowed to update the configuration of the skeleton. Contact is also modeled using hard constraints. Furthermore, the authors introduce a modified version of virtual coupling [6]. Perez et al. [30, 32] investigated the simulation of non-linear skin based on strain-limiting constraints and presented an efficient method for solving this kind of constrained problems [31].

Hirota et al. [14] published a method in which a detailed high-resolution skin model is simulated. In their approach, the skeleton is not simulated. Instead, the tracked hand configuration is directly applied to the skeleton, which acts as a boundary condition on the skin. This may result in too large stress and deformations in the skin since the simulation does not constrain the motion of the user's hand, nor compensate for large deviations from the constraints imposed by virtual objects. In contrast, the work in [17] deforms the skin using a global blending of the skin based on the configuration of the underlying skeleton. Here, no physically based simulation for the skeleton and skin is performed. However, on top of this, a local deformation is computed based on the penetration depth of the skin and a grasped object.

Our proposed hand simulation method differs from all previous work. It reaches an accuracy similar to constraint-based methods in the treatment of skeletal joints, frictional contact, and skin non-linearity, but it does so in a much more efficient manner, within a unified energy-minimization solver.

3 Soft Hand Simulation Model

The main philosophy behind our simulation method is to maximize stability by solving a unified energy minimization problem. The solution to this problem results in the velocities of the system's degrees of freedom. Constraints are modeled through *soft constraints* or penalty energies, instead of using *hard constrains* and solving a large constrained optimization problem. With soft constraints, joint and contact constraints are not solved exactly, but we show that the resulting inaccuracies are visually imperceptible. On the other hand, we avoid the computational cost of constrained optimization, and we guarantee that the simulation is executed with a deterministic cost on every frame, see Sect. 5.2. This feature is key for the robustness of our method. In the following sections we discuss the individual parts of the simulation method. We start by describing the overall dynamics solver, followed by details about the various energy terms.

3.1 Dynamics

Let us define generalized coordinates $\mathbf{q}$, which comprise soft tissue nodal coordinates $\mathbf{x}_t$, and linear and angular coordinates $\{\mathbf{x}_r, \theta_r\}$ of various rigid bodies, such as the hand's phalanxes and rigid objects in the virtual environment. Let us also define a potential energy U and a kinetic energy $T = \frac{1}{2}\dot{\mathbf{q}}^T\mathbf{M}\dot{\mathbf{q}}$, with $\mathbf{M}$ the mass matrix of the full system, built from the mass matrix $\mathbf{M}_t$ of the soft tissue and the masses $\{m_r\}$ and inertia tensors $\{\mathbf{I}_r\}$ of the various rigid bodies.

The equations of motion are given by the Euler-Lagrange equations

$$\frac{\partial}{\partial t}\left(\frac{\partial T}{\partial \dot{\mathbf{q}}}\right) - \frac{\partial T}{\partial \mathbf{q}} + \frac{\partial U}{\partial \mathbf{q}} = \mathbf{0}, \tag{1}$$

which, assuming a constant mass matrix during each time step boils down to

$$\frac{\partial}{\partial t}(\mathbf{M}\dot{\mathbf{q}}) + \frac{\partial U}{\partial \mathbf{q}} = \mathbf{M}\ddot{\mathbf{q}} + \frac{\partial U}{\partial \mathbf{q}} = \mathbf{0}. \tag{2}$$

Using a forward difference approximation of accelerations and velocities, we obtain the backward-Euler time-discretization of the equations of motion:

$$\mathbf{h}(\mathbf{q}) = \mathbf{M}\left(\frac{\mathbf{q} - \mathbf{q}^i - \Delta t\dot{\mathbf{q}}^i}{\Delta t^2}\right) + \frac{\partial U}{\partial \mathbf{q}}. \tag{3}$$

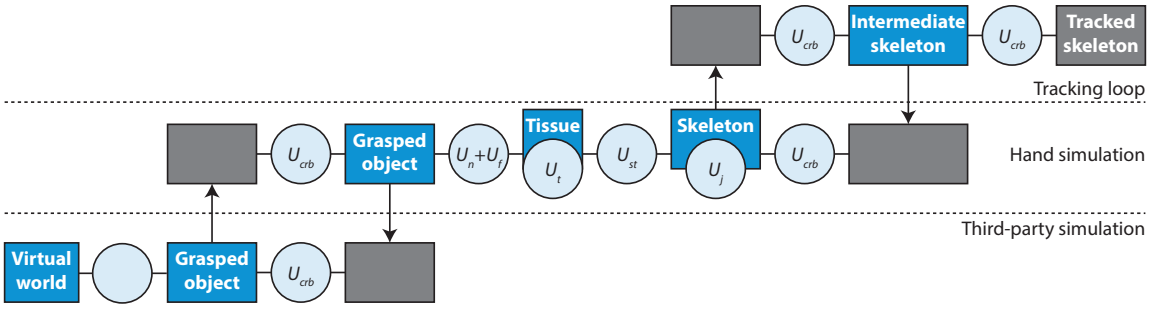


Figure 2: Overview of our hand simulation method. We simulate three concurrent loops, shown in three horizontal layers: a tracking loop, the full hand simulation, and a third-party simulation. The dark-blue boxes correspond to subsets of the simulation state $\mathbf{q}$, which are actually simulated; the light-blue circles correspond to energy terms; and the gray boxes correspond to synchronization copies of state subsets. We also depict the tracked skeleton using a gray box, because it is considered as a set of static rigid bodies in the simulation.

As demonstrated in [10, 25], and now applied to our hand model, the time-discrete equations of motion correspond to the optimality condition of an energy function $E(\mathbf{q}) = \frac{\Delta t^2}{2}\ddot{\mathbf{q}}^T\mathbf{M}\ddot{\mathbf{q}} + U$, with $\ddot{\mathbf{q}} = \frac{\mathbf{q}-\mathbf{q}^i-\Delta t\dot{\mathbf{q}}^i}{\Delta t^2}$. In other words, the equations of motion are the gradient of the energy, $\mathbf{h}(\mathbf{q}) = \frac{\partial E}{\partial \mathbf{q}}$.

Solving the minimization using Newton's method yields $\mathbf{h}(\mathbf{q}^{i+1}) = \mathbf{h}(\mathbf{q}^i) + \Delta t\frac{\partial \mathbf{h}}{\partial \mathbf{q}}\dot{\mathbf{q}}^{i+1} = \mathbf{0}$. And after substituting Eq. 3, we obtain the following linear system for each Newton iteration:

$$\left(\mathbf{M} + \Delta t^2\frac{\partial^2 U}{\partial \mathbf{q}^2}\right)\dot{\mathbf{q}}^{i+1} = \mathbf{M}\dot{\mathbf{q}}^i - \Delta t\frac{\partial U}{\partial \mathbf{q}}, \tag{4}$$

with $\frac{\partial^2 U}{\partial \mathbf{q}^2}$ the Hessian matrix of the potential energy in the system.

We solve for the unknown velocities $\dot{\mathbf{q}}^{i+1}$ using the Preconditioned Conjugate Gradient method, and the new positions are obtained simply as $\mathbf{q}^{i+1} = \mathbf{q}^i + \Delta t\dot{\mathbf{q}}^{i+1}$. In practice, to keep the simulation cost bounded, we execute only one Newton iteration per time step.

To fully exploit the energy minimization formulation, we model all internal forces and all interactions through potential energy terms added toward the total potential energy:

$$U = U_t + \sum U_j + \sum U_{st} + \sum\left(U_n + U_f\right) + \sum U_{crb}. \tag{5}$$

The various U_j terms model skeletal joints (Sec. 3.2), U_t models the internal forces of the soft tissue (Sec. 3.3), U_{st} accounts for the coupling between the hand's skeleton and the soft tissue (Sec. 3.4), U_n and U_f model respectively non-penetration and friction (Sec. 3.5), and the various U_{crb} terms denote coupling energies between rigid bodies, which cover both elastic tracking (Sec. 3.6) and coupling with an external physics engine (Sec. 3.7). By solving all parts together, the solution takes the (indirect) influence between different bodies into account, which maximizes the stability of the simulation.

Our method includes three concurrent simulation instances: the main simulation, a tracking loop, and the external physics engine. The state of the main simulation comprises the soft tissue, rigid phalanxes, and the object being manipulated. In addition, the main simulation takes as input other instances of the phalanxes and the manipulated object, copied from the tracking loop and the external engine respectively. The state of the tracking loop comprises the rigid phalanxes, and takes as input the tracked configuration and another instance of the phalanxes copied from the main simulation. Finally, the physics engine, in addition to its internal state, takes as input an instance of the manipulated object copied from the main simulation. The full structure of our method is depicted in Fig. 2.

3.2 Articulated skeleton

The overall motion of the simulated virtual hand is dictated by the dynamics of the underlying articulated skeleton, given by the bones and joints which form the hand. Articulated bodies are typically modeled through a set of rigid bodies connected by joints, defined as constraints [1]. This approach works well for articulated rigid bodies, but poses some problems when one wants to couple the rigid bodies with the deformable skin of the hand. This requires one to also include the dynamics of the deformable skin into the formulation of the constrained problem. Since this is computationally expensive for a reasonably complex deformable skin, we choose to use *soft constraints* instead of *hard constraints*.

For each joint, we define a potential energy U_j. Taking as example a spherical joint between two rigid bodies a and b, the energy is:

$$U_j = \frac{1}{2}k\left\|\mathbf{x}_a + \mathbf{R}_a\mathbf{r}_a - \mathbf{x}_b - \mathbf{R}_b\mathbf{r}_b\right\|^2, \tag{6}$$

where $\mathbf{x}_a$, $\mathbf{x}_b$, $\mathbf{R}_a$, and $\mathbf{R}_b$ are the positions and rotations of the bodies, and $\mathbf{r}_a$ and $\mathbf{r}_b$ are the vectors from the centers of mass to the location of the spherical joint, expressed in the local reference system of each body. Similar potential energies can be defined for other types of joints (e.g., hinge or universal joints), as well as for joint limits.

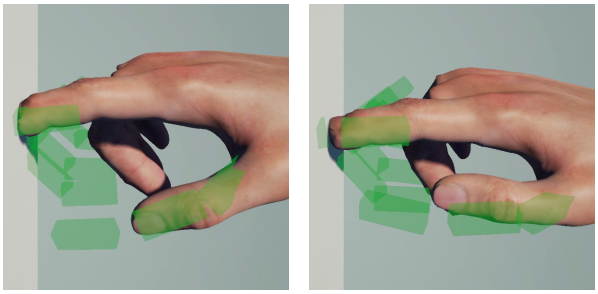
3.3 Modeling of nonlinear soft tissue

Human tissue is a typical example of nonlinear material. Under low pressure, the skin deforms easily. When pressure increases, the rate of deformation decreases and the material resists large pressure, see Fig. 3. In order to mimic the nonlinear behavior of skin, while retaining the computational simplicity of linear materials, we propose an extension to the co-rotational FEM method. This extension increases the underlying energy density when a certain threshold is reached. Our method builds on co-rotational FEM and only needs to compute the unrotated stiffness matrices once, which makes it attractive for interactive applications.

The potential energy of the elastic soft tissue can be described as

$$U_t = \int_\Omega W(\mathbf{F}(\mathbf{X}))d\mathbf{X}, \tag{7}$$

with W the *internal energy density function*, and $\mathbf{F}$ the deformation gradient at an undeformed point $\mathbf{X}$ within the material, see [40]. The co-rotational Finite Element Method (FEM) [27] discretizes the tissue using a tetrahedral mesh and computes an unrotated initial instance of the constant stiffness matrix $\mathbf{K}_0 = \frac{\partial^2 U_t}{\partial \mathbf{X}^2}$ per element. In every time step, $\mathbf{K}_0$ is rotated given the current relative rotation with respect to the initial configuration $\mathbf{X}$, i.e., $\mathbf{K} = \mathbf{R}\mathbf{K}_0\mathbf{R}^T$, and the full stiffness matrix is assembled. The rotations are obtained using a



(a) Linear skin (b) Nonlinear skin

Figure 3: Comparison between linear and nonlinear skin. The soft hand pushes against a static object, while the motion of the user (shown by the green bones) is unconstrained, resulting in a large stress and strain in the deformable skin. The soft linear model collapses, while the nonlinear model is able to resist high pressure.

Singular Value Decomposition (SVD) or *QR decomposition* of the deformation gradient $\mathbf{F}$ of the element.

While co-rotational FEM retains the benefits of linear materials, it fails to capture the nonlinear behavior of skin. A possible solution is to augment the co-rotational FEM method with strain-limiting constraints [13,30,31]. However, these methods are computationally demanding, as they require the solution to a constrained optimization problem per simulation time step.

Instead, we propose a method that does not add a large computational overhead to the running time of the simulation, yet it is able to mimic the nonlinear behavior of skin tissue. When the energy density function W of an element exceeds a certain threshold $W_{\max}$, we add a quickly growing energy term. Specifically, the energy density function becomes:

$$W_{\text{new}} = W + \frac{1}{2}k\left(\frac{W}{W_{\max}} - 1\right)^2, \tag{8}$$

with k an additional stiffness parameter. With co-rotational FEM, the underlying energy density can be obtained through the strain, stress and material parameters of each element (see [32]), but our nonlinear elasticity method is applicable to other constitutive models.

The gradient of the nonlinear energy is:

$$\frac{\partial W_{\text{new}}}{\partial \mathbf{x}} = \left(1 + \frac{k}{W_{\max}}\left(\frac{W}{W_{\max}} - 1\right)\right)\frac{\partial W}{\partial \mathbf{x}}. \tag{9}$$

And the Hessian is:

$$\frac{\partial^2 W_{\text{new}}}{\partial \mathbf{x}^2} = \left(1 + \frac{k}{W_{\max}}\left(\frac{W}{W_{\max}} - 1\right)\right)\frac{\partial^2 W}{\partial \mathbf{x}^2} + \frac{k}{W_{\max}^2}\left(\frac{\partial W}{\partial \mathbf{x}}\right)\left(\frac{\partial W}{\partial \mathbf{x}}\right)^T. \tag{10}$$

As it becomes evident, the per-element forces and stiffness matrix are easily obtained from the original forces and stiffness matrix. Moreover, if the original stiffness matrix is SPD, the new stiffness matrix is also SPD, as it is computed as a scaled version of the original stiffness matrix plus a term given by the outer product of the forces. The stiffness matrix is not continuous at the energy threshold, but this does not affect the solution of dynamics within each time step. Finally, the values of k and $W_{\max}$ must be chosen carefully to produce the desired nonlinear behavior.

3.4 Coupling between bones and tissue

We couple the soft tissue of the hand to the underlying skeleton by defining energy terms that penalize the deviation between the skin tetrahedral mesh and the skeletal rigid bodies. For each bone, we define a capsule, as shown in Fig. 4. If a mesh element e intersects the bone capsule, we sample the surface of the capsule, and average the positions of the samples that lie inside e to obtain a single representative point $\mathbf{x}_e$. This point can be defined kinematically as a function of the bone position $\mathbf{x}_r$ and orientation θ_r through its position $\mathbf{r}_e$ in the local reference system of the bone, and it can also be defined kinematically as a function of element nodes $\{\mathbf{x}_i\}$ through its barycentric coordinates $\{w_i\}$. For each element and intersected bone combination, we define the following energy:

$$U_{st} = \frac{1}{2}kN\left\|\mathbf{x}_r + \mathbf{R}_r\mathbf{r}_e - \sum_{i=1}^{4} w_i\mathbf{x}_i\right\|^2, \tag{11}$$

with N the number of bone samples that intersect with the mesh element. We determine the stiffness k as a fraction of a total coupling stiffness k_{total} as $k = k_{\text{total}}/N_{\text{total}}$, where N_{total} is the total number of samples on the bone capsule. Under uniform sampling of the bone capsule, this strategy results in a coupling which is proportional to the intersected area for each bone-element pair.

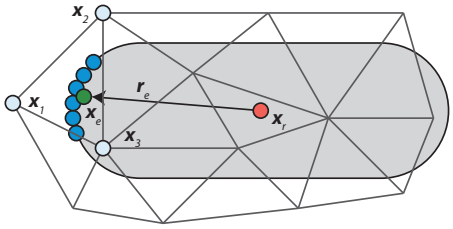


Figure 4: Coupling of one tetrahedron with a bone represented by its capsule (gray). The point $\mathbf{x}_e$ (green) is the average of the capsule surface samples intersecting a mesh element (dark blue).

3.5 Contact handling

For each colliding point on the skin $\mathbf{x}_n$, we model non-penetration and friction using energies that penalize the distance between $\mathbf{x}_n$ and an artificial anchor point $\mathbf{x}_a$ on the surface of the body being touched, see e.g. [39].

Contact

To accelerate collision detection, we compute a signed distance field for each object in the scene at initialization. At runtime, we check for collisions of vertices on the surface of the hand by querying the distance fields. When a surface vertex $\mathbf{x}_n$ collides, i.e., the distance is negative, we initialize the anchor $\mathbf{x}_a$ at the closest point on the surface of the touched object, as shown in Fig. 5a. We also compute the normal $\mathbf{n}$ at the anchor. Then, to model normal contact forces, we define a penalty energy

$$U_n = \frac{1}{2}wk_n\left\|\mathbf{n}\mathbf{n}^T(\mathbf{x}_a - \mathbf{x}_n)\right\|^2, \tag{12}$$

with k_n the contact stiffness and w a weighting value that is proportional to the area associated with the skin surface vertex.

Friction

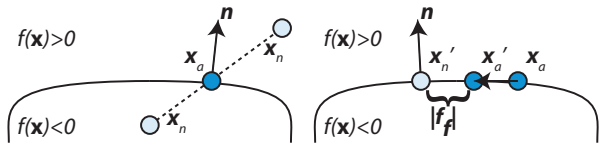
To model tangential (frictional) forces, we define a penalty energy between the deviation of the contact vertex $\mathbf{x}_n$ and the anchor $\mathbf{x}_a$ tangent to the surface:

$$U_f = \frac{1}{2}wk_f\left\|\left(\mathbf{I} - \mathbf{n}\mathbf{n}^T\right)(\mathbf{x}_a - \mathbf{x}_n)\right\|^2, \tag{13}$$

where k_f is a tangential stiffness.

By updating the anchor node, we determine the frictional behavior. If the anchor node is kept fixed, we achieve a stick friction regime. On the other hand, if the anchor node follows the contact vertex, we achieve a slip friction regime, with sliding friction. During stick conditions, friction acts as a conservative force, and it captures the tangential compliance of contact. During slip conditions, friction acts as a dissipative force.

To determine the friction regime, we use the Coulomb friction model. Let us define the normal force $\mathbf{f}_n = -\partial U_n / \partial \mathbf{x}_n$ and the friction force $\mathbf{f}_f = -\partial U_f / \partial \mathbf{x}_n$. Given a Coulomb friction coefficient μ, if $\|\mathbf{f}_f\| < \mu \|\mathbf{f}_n\|$, then the contact is in stick regime and the anchor is kept fixed. Otherwise, if $\|\mathbf{f}_f\| > \mu \|\mathbf{f}_n\|$, then the contact is in slip regime, we let the anchor slide tangent to the surface toward the contact vertex until the Coulomb condition holds, and we recompute the friction force. Fig. 5 depicts the computation of the anchor in stick and slip regimes.



(a) Configuration at first impact (b) Update of anchor

Figure 5: Our contact model. Left: a vertex $\mathbf{x}_n$ of the skin penetrates the object ($f(\mathbf{x}) < 0$). At the point of intersection, anchor $\mathbf{x}_a$ is initialized and the contact normal $\mathbf{n}$ is computed. Right: if kinetic friction applies, anchor $\mathbf{x}_a$ slides over the surface such that the corresponding friction force respects Coulomb's friction model.

3.6 Elastic tracking

We wish to control the simulated hand by tracking the motion of the user's real hand. While applying the motion of the tracked hand directly to the skeleton of the virtual hand may appear logical, this strategy suffers multiple problems, as shown in the accompanying video. Since the user's real hand is unconstrained, the forces on the grasped virtual object can grow rapidly. Moreover, if the tracking output is noisy or discontinuous, sudden displacements of the hand may also turn into excessive forces.

As an alternative to direct application of the tracked hand configuration, we implement a virtual coupling scheme [6] between the simulated hand and the tracked hand. Such virtual coupling absorbs discontinuities introduced by the tracking, and it also absorbs large deviations when the simulated hand is constrained by contact, as shown in Fig. 6. As a result, forces are always within bounds and the simulation is robust.

Layered setup

To maximize the stability of the tracking output fed to the simulation, we execute a simulation of the hand skeleton at a higher rate. This high-rate skeletal simulation produces the configuration of *intermediate bones*, taking as input the tracked hand configuration and the full skeletal simulation. On the other hand, in the full hand simulation, the tracked hand configuration is replaced by the intermediate bones, as shown in Fig. 2. In summary, we maintain three instances of skeletal representations:

1. Tracked configuration of the hand skeleton
2. Intermediate bones
3. Full articulated skeleton, with joints, coupled with the tissue

To ensure that the intermediate bones follow the tracked configuration, we model a coupling energy for each pair of corresponding bones. Given two rigid bodies a and b (i.e., a tracked bone and its corresponding intermediate bone), we define a coupling energy

$$U_{crb} = \frac{1}{2} k_x \|\mathbf{x}_a - \mathbf{x}_b\|^2 + \frac{1}{2} k_\theta \|\Delta\theta\|^2 \tag{14}$$

with $\mathbf{x}_a$ and $\mathbf{x}_b$ the positions of the two bones, and $\Delta\theta$ the axis-angle representation of their relative rotation. k_x and k_θ are, respectively, linear and angular coupling stiffness values.

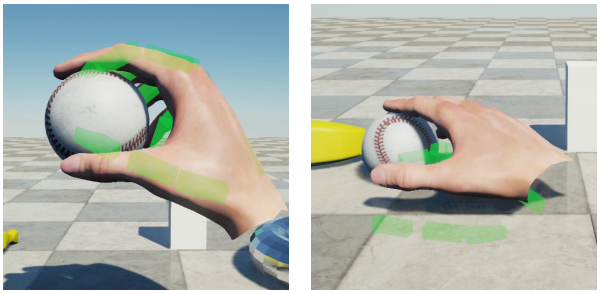


Figure 6: Examples of elastic tracking: The user's hand is shown in green. The images show the virtual hand while squeezing and pushing on a baseball. The motion of the virtual hand is obstructed in a realistic and natural way.

Coupling between two skeleton simulations

Since the high-rate intermediate simulation and the full simulation take place at different update rates, the coupling between the intermediate bones and the full skeleton requires a different treatment. The high-rate simulation maintains a copy of the full skeleton, which is updated after every full simulation step. Similarly, the full simulation maintains copies of the intermediate bones, which are updated after every high-rate simulation step.

Both in the high-rate simulation and the full simulation, we model the coupling between corresponding bones using the same coupling energy as in Eq. 14. However, for the coupling between phalanxes, we set $k_x = 0$. In practice, this amounts to coupling both translation and orientation for the palm, and only orientations for the phalanxes. We observe that, in this way, hand poses propagate more naturally through skeletal joints.

In the high-rate simulation, we model the bones as disconnected rigid bodies. Since the tracked configuration respects physical joint constraints, with disconnected bodies we obtain sufficient accuracy under much lower simulation cost. Note that, in the high-rate loop, the copy of the full skeleton is maintained constant, and the simulation boils down to the computation of rigid body dynamics for 16 rigid bodies. As a result, the cost of the high-rate simulation is negligible in contrast to the full simulation, and we succeed to support fast motion of the user's hand robustly, as shown in the accompanying video.

3.7 Coupling with external physics engine

Our method can be used together with third-party physics engines. We use our method to simulate the hand and the object grasped by the hand, but we rely on a third-party physics engine to simulate the rest of the virtual world.

To connect our hand simulation and the physics engine, we simulate two instances of the grasped object, as shown in Fig. 2, and we couple them using the rigid-body coupling energy given in Eq. 14. To implement this coupling, within each simulation we maintain a copy of the other instance.

However, a straightforward copy of body states across simulations makes the coupling explicit, even though the integration method we use is implicit, and the simulations are prone to instabilities. Instead, to improve accuracy and stability, we use higher order approximations of the positions and orientations of the copies. In our hand simulation, let us denote the local instance of the grasped object as a, and the instance copied from the physics engine as b. In addition to the position and orientation of b, we also copy from the physics engine its velocities and accelerations. Then, prior to a simulation step, we approximate the position and orientation of b assuming constant acceleration since the previous update. In some physics engines, velocities and accelerations might be readily available or might be computed from the applied force and torque. If this is not possible, we approximate them using finite differences.

Thanks to our coupling method, we achieve a complete two-way coupling between the hand and the virtual world. The hand simulation is able to respond to collisions of the grasped object with other objects in the virtual world, while the objects in the virtual world can be moved indirectly by our simulated hand through the grasped object. See the accompanying video and Fig. 7 for an example. Furthermore, if the scales between the two simulations are different, the inertia tensor, linear velocity and acceleration need to be scaled. See [11] for more information about scaling properties between two simulations. The mass of the objects in both simulations are set to the same value.

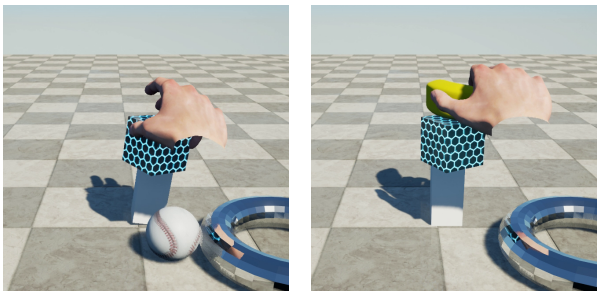


Figure 7: Thanks to the coupling between our simulation and the external physics engine, the user is able to interact with the full virtual environment. Our simulation deals only with the motion of the hand and the grasped object.

4 Integration in a Game Engine

As mentioned in the introduction, one major advantage of our simulation method is that it enables seamless integration with off-the-shelf physics engines. Building on this feature, we have encapsulated our hand simulation method within CLAP, a library that simplifies the integration with other modules. In particular, CLAP constitutes an asset of a game engine, and the soft hand model can be easily integrated in a virtual environment. CLAP is a modular library for hand simulation, which allows easy integration of different types of tracking hardware, and it is also able to drive various types of haptics and tactile devices through device-specific *drivers* or *modules*. CLAP also provides an interface for third-party simulation-environments, like Unity or Unreal Engine, to share information about the state of objects that are allowed to interact with our simulation. Furthermore, these environments are used for displaying the state of the simulated hand. Thanks to the modular setup, users may develop customized drivers for their hardware, with easy integration with existing software. Fig. 8 shows a schematic overview of the CLAP library.

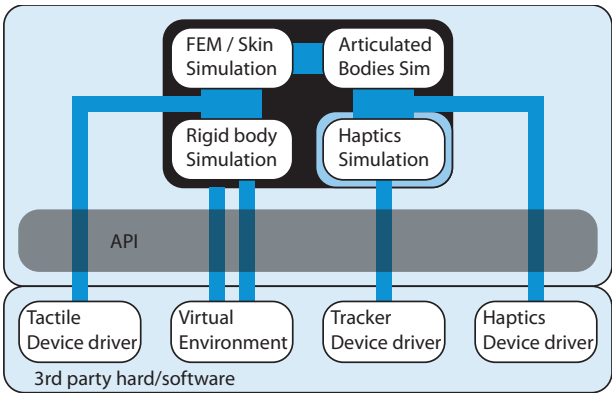


Figure 8: Schematic overview of the CLAP library.

5 Results

5.1 Implementation Details

In all examples, we discretize the soft tissue using a tetrahedral mesh with 856 elements, created using TetGen [36]. The deformation of the tetrahedral mesh is interpolated to an embedded high-resolution triangle mesh with 23640 triangles representing the surface of the hand. For the soft tissue model, we have set Young's modulus to 10^5 Pa and Poisson's ration to 0.33.

For the various energy terms in the model, we have used the following settings. In the coupling between skeleton and tissue, we have used a total stiffness $k_{\text{total}} = 3.2 \times 10^6$ N/m. For the skeletal joints, we have used a high stiffness of 2×10^5 N/m to minimize constraint deviation. For the nonlinear skin mode, we have set the maximum energy density $W_{\max}$ to 2500 J/m^3, and the nonlinear energy stiffness to 1000 J/m^3. For the coupling between the tracked and simulated hands, we have set the linear and angular stiffness values to 50 N/m and 200 Nm/rad respectively. For the coupling with the third-party engine, we have set them to 170 N/m and 70 Nm/rad respectively.

All graspable rigid objects in the virtual world had a mass of 0.1 kg, with a Coulomb friction coefficient $\mu = 0.5$. In addition, at initialization we compute signed distance fields of 64^3 voxels with an additional narrow-band of 8 voxels outside the object, which take 3 MB of memory on average. For complex shapes, this could grow up to 6 MB.

We have integrated our hand simulation in Unreal Engine 4, using NVIDIA PhysX 3.3 as third-party physics engine. In our examples, the third-party simulation loop runs at a rate between 60 and 120 Hz. For hand tracking, we have used LeapMotion, and the tracking loop runs at a fixed rate of 250 Hz. Finally, we have used Eigen to implement the linear algebra operations in the simulation. In each simulation step, we solve the linear system in Eq. 4 using the Conjugate Gradient method with warm start, a Jacobi preconditioner and a tolerance of 10^{-5}.

5.2 Timings

We carried out all experiments on an Intel Core i7-5820K CPU, with the full hand simulation executed in a single core. Fig. 9 shows timing results of a simulation in which the user grasps, squeezes, releases and touches the ball object shown in Fig. 6. Overall, we obtain a stable simulation frame rate between 45 and 65 steps per second. At the beginning of the simulation, before the tracker detects the hand, the frame rate reaches 90 steps per second, and suddenly drops to about 50 steps per second, with a fast and sudden motion of the simulated hand. Thanks to elastic tracking, the simulation handles this event robustly. In normal operation mode, the simulation cost is split almost evenly between setting up and solving the system

in Eq. 4. Each piece takes approximately 10 ms and the dominant factor affecting the cost is the resolution of the FEM mesh. When the user firmly grasps an object, the solver requires slightly more time to converge due to an increase in contacts and energy terms. The cost of collision detection, however, remains negligible under 0.7 ms, thanks to the use of precomputed distance fields.

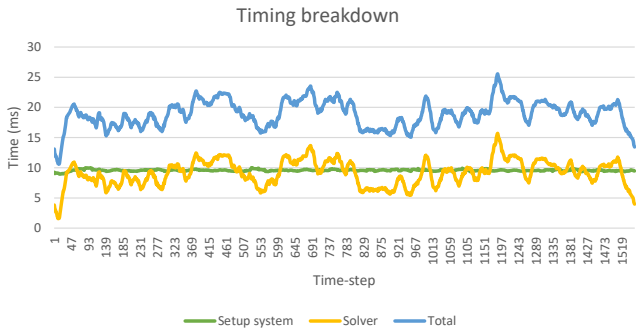


Figure 9: Breakdown of timings of our simulation. The simulation cost is split almost evenly between system setup and system solve.

5.3 Stability

Thanks to the coupling between the simulated skeleton and the tracked hand, we obtain a stable and smooth motion of the simulated hand. The simulated system consists of multiple energy terms, but since all the energy terms are minimized at once, we reach a stable result. The inclusion of contact and friction as part of the energy terms ensures that we also achieve very stable grasping of arbitrary objects. In the accompanying video and Fig. 6, we show examples where the user smashes an object on the floor while holding it in the hand, builds a stack of arbitrary objects, or interacts naturally with the objects. While these scenarios are challenging for other methods, we show the robustness of our approach. In the same video, we compare our method with a direct application of the tracked hand configuration to the skeleton of the simulated hand. In this case, excessive forces are not absorbed and the tissue deforms unrealistically, while the motion of the hand and fingers is noisy and sometimes discontinuous. When an object is pushed on the ground, the system starts to vibrate due to the high stiffness of the contact energies.

5.4 Skin deformation

Fig. 3 shows the differences between the linear and nonlinear skin models presented in Sect. 3.3. The deformable hand is pushed against a static object by tracking the motion of the user's hand. Note that the tracked hand is shown in green, and it has a similar deviation with respect to the simulated hand in both cases. In Fig. 3a, the linear FEM model cannot resist the pressure generated by the difference between the tracked and simulated skeleton. In Fig. 3b, on the other hand, the nonlinear model performs well under the same conditions.

5.5 Coupling with existing physics engines

Fig. 7 shows snapshots of our simulation coupled with the NVIDIA PhysX engine. Thanks to the coupling described in Sect. 3.7, the user is able to interact with the virtual environment that is simulated in a different simulation loop. Please note that instances of objects simulated in our simulation are not aware of each other since no collisions are detected between the objects in our simulation. However, collisions are detected in the third-party simulation and energies between the instances are exchanged. Therefore, the motion of the virtual hand is effectively influenced by these collisions.

6 Discussion and Future Work

Thanks to the simulation method described in this paper, we are able to naturally interact with a virtual environment using a robust simulation of a virtual hand. We show that a coarse tetrahedral mesh (e.g., in contrast to the one of [14]) is sufficient for obtaining smooth and natural interaction. The key aspect is to simulate the full skeletal dynamics of the hand, to model interactions using energy potentials, and to solve dynamics as a single energy minimization problem.

Our simulation method is not free of limitations. In order to maintain a deterministic simulation cost, we have made several strategic design decisions that currently limit the range of available interactions. First, we do not support self-collisions of the hand. As our fast collision detection relies on precomputed distance fields, we do not support contact between deformable objects or the within the hand. This limitation could be relaxed by using fast distance field computation methods (e.g., [12]). Second, the object being touched by the hand must be rigid. Our current solver relies on a fixed state size and a rigid-body coupling between the main simulation and the third-party engine. Adding the ability to touch deformable objects would require more versatile state management, support of deformable objects by the third-party engine, and coupling between deformable objects. Third, the user is limited to touching only one object at a time. In this way, we bound the cost of the hand simulation loop and the complexity of the coupling to the third-party engine. Fourth, modeling interactions through coupling potentials may require parameter tuning in certain cases. In particular, lifting heavy objects might require stiffer couplings. For ordinary objects lighter than 0.5 kg, we have observed no issues with the default parameters.

In addition to relaxing limitations, there are further ways in which we could extend the features of our method. Some applications might require a higher-resolution tetrahedral mesh for the hand's soft tissue. For instance, currently the nails may deform under high-pressure contact, and interactions with sharp edges are not accurately. Porting our simulation method to the GPU might enable a finer tetrahedralization. Our hand model is also limited to be right-handed and of fixed size. With the addition of a parameterized hand model with variable phalanx length, width, gender, and skin color, we can adapt the model to each particular user, making tracking more accurate and the interaction experience more personal. Finally, another interesting extension is the support for two-handed interaction.

Our work is complementary to haptic and tactile rendering, and our simulation model could serve as an excellent foundation for the computation of haptic and tactile output. The addition of haptic feedback could make interactions even more natural by restricting the motion of the user. Currently, the user's motion is unconstrained, and the tracked hand postures may be difficult to convey to the simulation when the user largely penetrates virtual constraints. [24] used an earlier version of our simulation method for driving an underactuated haptic device and thus constrain the motion of the user's fingers.

To conclude, in future work we would like to conduct a thorough user study and further validate our model. The current version of the CLAP library is available for Unreal Engine and will support Unity in the near future.

Acknowledgments

The authors wish to thank the anonymous reviewers for their helpful comments, as well as the members of the MSLab at URJC Madrid, in particular Juan José Casafranca for his help with generating the tetrahedral meshes and rigging the hand model. This project was partially supported by grants from the EU FP7 (project no. 601165 WEARHAP) and the Spanish Ministry of Economy (TIN2015-70799-R and TIN2017-82091-ERC).

REFERENCES

[1] D. Baraff. Linear-time dynamics using Lagrange multipliers. *Proc. of ACM SIGGRAPH*, 1996.

[2] F. Barbagli, A. Frisoli, K. Salisbury, and M. Bergamasco. Simulating human fingers: a soft finger proxy model and algorithm. In *HAPTICS '04. Proc. 12th International Symposium on*, 2004.

[3] C. W. Borst and A. P. Indugula. Realistic virtual grasping. In *Proc. of IEEE Virtual Reality Conference*, 2005.

[4] M. Catalano, G. Grioli, E. Farnioli, A. Serio, C. Piazza, and A. Bicchi. Adaptive synergies for the design and control of the pisa/iit softhand. *The International Journal of Robotics Research*, 33(5):768–782, 2014.

[5] M. Ciocarlie, C. Lackner, and P. Allen. Soft finger model with adaptive contact geometry for grasping and manipulation tasks. In *World Haptics Conference*, 2007.

[6] J. E. Colgate, M. C. Stanley, and J. M. Brown. Issues in the haptic display of tool use. *Proc. of IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages pp. 140–145, 1995.

[7] C. Duriez, H. Courtecuisse, J.-P. d. Plata Alcalde, and P.-J. Bensoussan. Contact skinning. *Eurographics conference (short paper)*, 2008.

[8] A. Frisoli, F. Barbagli, E. Ruffaldi, K. Salisbury, and M. Bergamasco. A limit-curve based soft finger god-object algorithm. In *Haptic Interfaces for Virtual Environment and Teleoperator Systems, 14th Symposium on*, 2006.

[9] C. Garre, F. Hernandez, A. Gracia, and M. A. Otaduy. Interactive simulation of a deformable hand for haptic rendering. In *Proc. of World Haptics Conference*, 2011.

[10] T. F. Gast, C. Schroeder, A. Stomakhin, C. Jiang, and J. M. Teran. Optimization integrator for large time steps. *IEEE Transactions on Visualization and Computer Graphics*, 21(10):1103–1115, 2015.

[11] L. Glondu, M. Marchal, and G. Dumont. Evaluation of Physical Simulation Libraries for Haptic Rendering of Contacts between Rigid Bodies. In *ASME 2010 World Conference on Innovative Virtual Reality (WINVR2010)*, 2010.

[12] L. Glondu, S. C. Schvartzman, M. Marchal, G. Dumont, and M. A. Otaduy. Fast collision detection for fracturing rigid bodies. *IEEE Transactions on Visualization and Computer Graphics*, 20(1):30–41, 2014.

[13] F. Hernandez, G. Cirio, A. G. Perez, and M. A. Otaduy. Anisotropic strain limiting. In *Proc. of CEIG*, 2013.

[14] K. Hirota and K. Tagaw. Interaction with virtual object using deformable hand. *2016 IEEE Virtual Reality (VR)*, 00:49–56, 2016.

[15] J. Jacobs and B. Froehlich. A soft hand model for physically-based manipulation of virtual objects. In *2011 IEEE Virtual Reality Conference (VR)*, pages 11–18, Mar. 2011.

[16] J. Jacobs, M. Stengel, and B. Froehlich. A generalized god-object method for plausible finger-based interactions in virtual environments. In *2012 IEEE Symposium on 3D User Interfaces (3DUI)*, pages 43–51, Mar. 2012.

[17] J. Kim and J. Park. Physics-based hand interaction with virtual objects. In *IEEE International Conference on Robotics and Automation, ICRA*, pages 3814–3819, 2015.

[18] P. G. Kry, D. L. James, and D. K. Pai. Eigenskin: Real time large deformation character skinning in hardware. In *ACM SIGGRAPH Symp. on Computer Animation*, pages 153–160, July 2002.

[19] P. G. Kry and D. K. Pai. Interaction capture and synthesis. *ACM Transactions on Graphics*, 25(3):872–880, July 2006.

[20] T. Kurihara and N. Miyata. Modeling deformable human hands from medical images. In *2004 ACM SIGGRAPH / Eurographics Symp. on Computer Animation*, pages 355–363, July 2004.

[21] B. León, S. Ulbrich, R. Diankov, G. Puche, M. Przybylski, A. Morales, T. Asfour, S. Moisio, J. Bohg, J. Kuffner, and R. Dillmann. *OpenGRASP: A Toolkit for Robot Grasping Simulation*, pages 109–120. 2010.

[22] Y. Li, J. L. Fu, and N. S. Pollard. Data-driven grasp synthesis using shape matching and task-based pruning. *IEEE Transactions on Visualization and Computer Graphics*, 13(4):732–747, July/Aug. 2007.

[23] C. K. Liu. Synthesis of interactive hand manipulation. In *2008 ACM SIGGRAPH / Eurographics Symp. on Computer Animation*, pages 163–172, July 2008.

[24] D. Lobo, M. Saraç, M. Verschoor, M. Solazzi, A. Frisoli, and M. A. Otaduy. Proxy-based haptic rendering for underactuated haptic devices. In *2017 IEEE World Haptics Conference (WHC)*, pages 48–53, June 2017.

[25] S. Martin, B. Thomaszewski, E. Grinspun, and M. Gross. Example-based elastic materials. *ACM Trans. Graph.*, 30(4):72:1–72:8, July 2011.

[26] A. Miller and P. K. Allen. Graspit!: A Versatile Simulator for Robotic Grasping. *IEEE Robotics and Automation Magazine*, 11(4), 2004.

[27] M. Müller and M. H. Gross. Interactive virtual materials. In *Graphics Interface 2004*, pages 239–246, May 2004.

[28] I. Oikonomidis, N. Kyriazis, and A. Argyros. Efficient model-based 3d tracking of hand articulations using kinect. In *Proc. of the British Machine Vision Conference*, pages 101.1–101.11, 2011.

[29] R. Ott, F. Vexo, and D. Thalmann. Two-handed haptic manipulation for CAD and VR applications. *Computer Aided Design & Applications*, 7(1), 2010.

[30] A. G. Perez, G. Cirio, F. Hernandez, C. Garre, and M. A. Otaduy. Strain limiting for soft finger contact simulation. In *Proc. of IEEE World Haptics Conference*, 2013.

[31] A. G. Perez, G. Cirio, D. Lobo, F. Chinello, D. Prattichizzo, and M. A. Otaduy. Efficient nonlinear skin simulation for multi-finger tactile rendering. In *Proc. of Haptics Symposium*. IEEE, 2016.

[32] J. Perez, A. G. Perez, and M. A. Otaduy. Simulation of hyperelastic materials using energy constraints. In *Proc. of CEIG*, 2013.

[33] N. S. Pollard and V. B. Zordan. Physically based grasping control from example. In *2005 ACM SIGGRAPH / Eurographics Symp. on Computer Animation*, pages 311–318, July 2005.

[34] M. Pouliquen, C. Duriez, C. Andriot, A. Bernard, L. Chodorge, and F. Gosselin. Real-time finite element finger pinch grasp simulation. In *Eurohaptics Conference, 2005 and Symposium on Haptic Interfaces for Virtual Environment and Teleoperator Systems, 2005. World Haptics 2005. First Joint*, pages 323–328, Mar. 2005.

[35] A. R. Rivers and D. L. James. FastLSM: Fast lattice shape matching for robust real-time deformation. *ACM Transactions on Graphics (Proc. of SIGGRAPH 2007)*, 2007.

[36] H. Si. Tetgen, a delaunay-based quality tetrahedral mesh generator. *ACM Trans. Math. Softw.*, 41(2):11:1–11:36, Feb. 2015.

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

[38] A. Talvas, M. Marchal, C. Duriez, and M. A. Otaduy. Aggregate constraints for virtual manipulation with soft fingers. *IEEE Transactions on Visualization and Computer Graphics*, 21(4):452–461, 2015.

[39] M. Tang, D. Manocha, M. A. Otaduy, and R. Tong. Continuous Penalty Forces. *ACMTOGSIGGRAPH*, 31(4):107:1–107:9, July 2012.

[40] J. Teran, S. Blemker, V. N. T. Hing, and R. Fedkiw. Finite volume methods for the simulation of skeletal muscle. In *SIGGRAPH/Eurographics Symp. on Computer Animation*, pages 68–74, 2003.

[41] N. Vahrenkamp, M. Kröhnert, S. Ulbrich, T. Asfour, G. Metta, R. Dillmann, and G. Sandini. *Simox: A Robotics Toolbox for Simulation, Motion and Grasp Planning*, pages 585–594. 2013.

[42] C. Zilles and J. Salisbury. A constraint-based god-object method for haptic display. In *Intelligent Robots and Systems, IEEE/RSJ International Conference on*, volume 3, page 3146. IEEE Computer Society, 1995.