A Soft Hand Model for Physically-based Manipulation of Virtual Objects

Jan Jacobs*
Group Research Virtual Technologies
Volkswagen AG

Bernd Froehlich†
Virtual Reality Systems Group
Bauhaus-Universität Weimar

ABSTRACT

We developed a new hand model for increasing the robustness of finger-based manipulations of virtual objects. Each phalanx of our hand model consists of a number of deformable soft bodies, which dynamically adapt to the shape of grasped objects based on the applied forces. Stronger forces directly result in larger contact areas, which increase the friction between hand and object as would occur in reality. For a robust collision-based soft body simulation, we extended the lattice-shape matching algorithm to work with adaptive stiffness values, which are dynamically derived from force and velocity thresholds.

Our implementation demonstrates that this approach allows very precise and robust grasping, manipulation and releasing of virtual objects and performs in real-time for a variety of complex scenarios. Additionally, laborious tuning of object and friction parameters is not necessary for the wide range of objects that we typically grasp with our hands.

Index Terms: I.3.5 [Computer Graphics]: Computational Geometry and Object Modeling—Physically based modeling; I.3.6 [Computer Graphics]: Methodology and Techniques—Interaction techniques; I.3.7 [Computer Graphics]: Three-Dimensional Graphics and Realism—Virtual reality; H.5.2 [Information Interfaces and Presentation]: User Interfaces—direct manipulation;

1 INTRODUCTION

The integration of physical behavior has significantly increased the quality of games and virtual environments overall. Consequently, the interaction with simulated objects also needs to occur on a physical basis. This turns out to be quite challenging, since the representation of a user in the virtual world needs to be physically modeled to achieve a realistic interaction between user and virtual objects. One particular difficulty is the modeling of the finely articulated human hand to enable finger-based interaction. Pioneering work by Borst et al. [1] relies on a hand model constructed from rigid bodies. While this approach was the first to demonstrate the potential of finger-based physical interaction, it could not correctly consider friction between fingers and virtual objects due to the rigid body approach and thus required careful tuning of parameters for a reasonably stable interaction.

We developed a new hand model for increasing the robustness of finger-based manipulations of virtual objects. Our approach is based on a soft body model for each finger phalanx to enable pressure-based deformation of the soft finger contact areas when object surfaces are encountered. Thus, friction is naturally increased by higher pressure and results in a firmer grip of an object. Realistic friction handling of the deformable objects is realized by an extended version of the Fast Lattice Shape Matching *(FastLSM)* algorithm [14]. Our extensions allow us to consider dynamically changing stiffness parameters based on force and velocity computations and lead to a robust grasping behavior.

Our work is motivated by the vision to realize a fully functional virtual car model, wherein every part of the car is physically modeled and simulated. Such a car model allows many more aspects to be virtually assessed as compared to today's reduced mockups. In addition, the collaborative discussion and evaluation of car functionalities (see e.g. [15]) require a system that can simultaneously handle multiple user inputs. As already noted in [6], physics provides this capability for free, since the superposition of applied forces is a basic ingredient in any physics simulation. While these systems are getting increasingly more robust and allow for the handling of thousands of ideally convex objects, they are not at the point where each knob and screw of a car can be physically simulated. Nevertheless, if parts of a car can be handled (e.g. such as the trunk or the engine) and manipulated by the hands and fingers of multiple users, a first and important step has been achieved.

The main contribution of this work is a new hand model based on a hybrid approach using soft bodies coupled with rigid bodies, which simulate the back of the hand. Furthermore, our extended version of the *FastLSM* algorithm considers adaptive stiffness values, which are a requirement for the robustness of our approach and make the *FastLSM* algorithm applicable to a much larger class of problems. The dynamic adaptation of the stiffness values also leads to a much simplified configuration of object and friction parameters for a wide variety of scenarios. Our implementation shows that the system allows for very precise and robust finger-based grasping, manipulation and releasing of virtual objects. The CPU implementation performs in real-time for various complex scenarios and could be directly applied to finger-based manipulations in many application areas.

2 RELATED WORK

A direct and robust finger-based manipulation relies on three major issues: stable grasping of objects, robust manipulation and controlled releasing of objects. In general, there are two common ways to achieve these goals: grasping through heuristics and collision-based physical simulations.

The use of a heuristic classification of the hand pose for the detection of valid grasps can be sufficient (e.g.[9]), since grasping of objects is performed by just a few different hand poses [22]. Moehring et al. [10] showed that a heuristics-based approach could be combined with a pseudo-physical simulation to generate plausible object movement. They focus on the particular manipulation of constrained objects by implementing a set of rules for each constraint type. The realism of the interaction is limited according to the implemented rules and remains mostly plausible, but it lacks general physical correctness. Heuristics-based approaches for direct finger-based manipulation have in common that they enable hand-object interactions while object-object interactions are hard to plausibly simulate without physics and thus are mostly ignored.

There are many techniques for offline simulation of virtual grasping for use with animated characters, which do not have the strong real-time requirements as virtual grasping controlled by a user does. Li et al. [7] used a database of predefined hand poses to identify an ideal grasping pose. Their algorithm returns a set of

*e-mail: jan.jacobs@volkswagen.de
†e-mail: bernd.froehlich@uni-weimar.de

suitable hand poses and the animator has to choose the best matching one. Ciocarlie et al. [2] developed an approach based on finite elements to simulate the complex nature of soft finger-object contacts. The high computational requirements of this approach make its application for interactive virtual environments difficult.

Froehlich et al. [6] showed that the integration of a physical simulation into interactive virtual environments allows complex assembly tasks involving multiple hands and users to be supported. This new user interface paradigm exploited the inherent properties of physical simulations–in particular the superposition of applied forces and the consideration of constraints. Their interaction was based on connecting a set of springs from the user's hand to the physically simulated virtual object. Borst et al. [1] significantly extended this concept by creating a rigid body hand representation for a direct interaction based on collision forces. A stable grasping could be established by assigning high friction values to the rigid-body elements. However, this could result in a certain stickiness and could make the releasing of objects awkward. The hand representation was coupled to the user's input by a spring-damping system. Both the choice of friction parameters and spring-damping coefficients made the approach fragile and only careful manual parameter optimization led to stable interaction results.

Physics simulations rely on a basic friction model to simulate object-object interaction for which it is mostly suitable, but it is of limited use for precise and robust object manipulation. Duriez et al. [5] addressed this problem by directly calculating friction at skin level. They developed a skinning method based on a skeleton-driven hand motion. Their quite realistic deformable hand model is used for grasping rigid body objects. However, they only achieved interactive frame rates for scenes with very low complexity (1500 triangles). Our goal is to use a real-time soft body approach to provide an implicit friction model which relies on a dynamically varying contact area and adaptive finger phalanx stiffness.

3 Real World Grasping

Grasp stability in the real world largely depends on the applied contact force in proportion to the object's weight. When touching a surface, the contact surface increases and thus a higher amount of friction is applied between finger and object (see Figure 1). As the contact force increases, simultaneously the skin's surface gets equally stiffer. We demonstrate how this behavior could also be transmitted to virtual grasping.

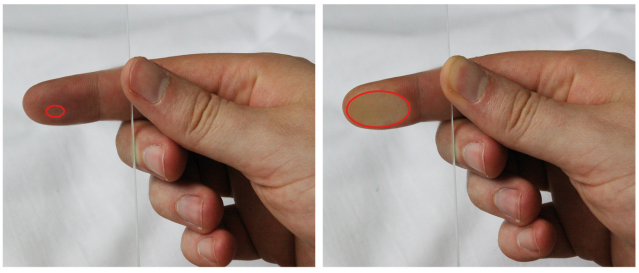

Figure 1: Increasing contact area with increasing contact force. left: loose touch. right: strong pressure between finger and a pane of glass. The red ellipse indicates the contact area in either case.

It is obvious that a deformable finger model should be preferred over a rigid body configuration in order to achieve this effect. However, just the introduction of soft bodies as finger phalanxes is not sufficient. If overly floppy soft bodies are used, the user will not be able to transfer the necessary force to attain stable grasping. A more rigid model does not allow for adaption to the object's surface. Therefore, a model for the realistic and dynamic approximation of the fingers' stiffness behavior is most important to allow a solid adaption of the contact area. As a result, we enable the user to realistically grasp and manipulate objects while maintaining a high contact force to avoid slipping.

4 Lattice Shape Matching

Our approach is based on the *FastLSM* algorithm introduced by Rivers et al. [14]. It is able to simulate both stiff and soft models at similar costs. In comparison to the original shape matching algorithm of Mueller et al. [11], *FastLSM* mainly improves the performance and controllability of the deformation of arbitrary geometry. Ohta et al. [12] showed that the shape matching algorithm of Mueller et al. is suitable to simulate the dynamic adaptation of stiffness at run time. This kind of versatility is also possible with our extended version of the *FastLSM* approach, which is at the same time computationally more efficient.

FastLSM voxelizes any geometry into a uniform lattice of cubic cells. This original lattice provides the rest configuration of the individual grid points. Their positions are denoted by $\mathbf{x}_i^0$. A weight m_i is assigned to each grid point i, which is referred to as a particle. If external forces $\mathbf{f}_{ext}$ act on these particles they will deform the lattice and the original surface geometry is deformed with the lattice. The idea now is that the particles are always seeking their rest position.

For a more sophisticated deformation model, several particles are grouped together in overlapping regions. Thus, particles influence each other's motion. Now, instead of moving towards their rest position, a target position $\mathbf{g}_i$ for each particle is calculated (Equation 1). The calculation of $\mathbf{g}_i$ considers the influence of other particles and defines the position each particle should move to. Therefore, at each time step, each region r finds the best rigid body transformation to match the initial configuration of its particles. This is done by calculating a least-squares transformation T_r. This computationally expensive calculation can be accelerated to be executed in constant time by using a fast summation operator F, the details of which can be found in [14]. In Equation 1, R_i denotes the list of particles that influence the particle i.

$$\mathbf{g}_i = \frac{1}{|R_i|} \underset{r \in R_i}{F} \{T_r\} \mathbf{x}_i^0 \tag{1}$$

Rivers et al. [14] introduced a parameter w, which controls the overlapping amount of the regions and thus the stiffness of the lattice. Figure 2 illustrates how the stiffness depends on the size and overlap of the regions. A fairly soft behavior could be achieved by using sparse overlapping regions (small w), and therefore regions that barely influence adjacent regions. Using a large w results in stiff behavior since the particles are reaching their target positions much faster. This behavior is pivotal to our approach in that it aids us in the development and ability to account for the dynamically adapting stiffnesses.

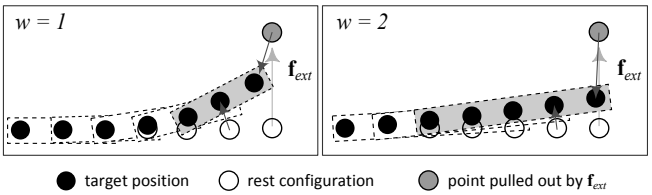

Figure 2: Small regions result in little overlap and soft behavior. Increasing region size and overlap increases the stiffness.

The *FastLSM* approach uses an Euler integration to simulate the movement of the particles towards their target positions. For this, Equations 2 and 3 are used to calculate the actual velocities $\mathbf{v}_i$ and

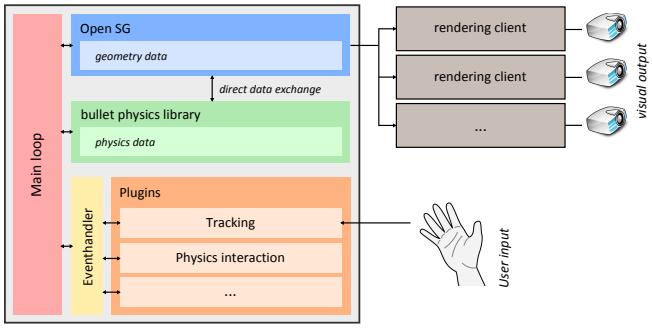

Figure 3: System architecture for direct, collision-based interaction in an immersive environment.

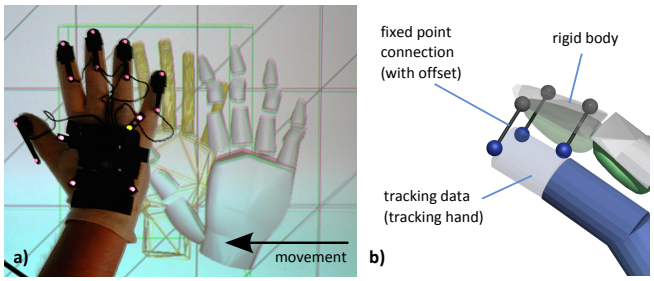

Figure 4: (a) Tracking latency between user's hand (left) tracked hand (middle) and physics hand (right). b) Fixed point coupling between tracked hand and physics hand.

positions $\mathbf{x}_i$ of the particles for each integration step h. Further details about the integration scheme can be found in [11].

$$\mathbf{v}_i(t+h) = \mathbf{v}_i(t) + \frac{\mathbf{g}_i(t) - \mathbf{x}_i(t)}{h} + h\frac{\mathbf{f}_{ext}(t)}{m_i} \quad (2)$$

$$\mathbf{x}_i(t+h) = \mathbf{x}_i(t) + h\,\mathbf{v}_i(t+h) \quad (3)$$

Steinemann et al. [16] proposed hierarchical voxelization to handle irregular meshes with a smaller amount of resulting particles. Furthermore, they described how different stiffness values could be applied for different regions of a body's shape. Since we use quite simple shapes for representing our hand model, we prefer a regular voxelization. Using *FastLSM* approach we found it to be very reliable and robust with convincing simulation results as a basis for our hand-model.

5 System Design

The following describes the main components of our approach. The software architecture shows the integration of our approach in a scenegraph system. Optical finger tracking ensures precise information about the pose of each finger. Our virtual hand model consists of a rigid skeleton and soft finger pads. The consideration of collisions and the use of a physical simulation warrant computation of physically realistic multi-handed and multi-user interactions. The physics simulation uses simplified proxy objects to guarantee interactive response times.

5.1 Software Architecture

We integrated our approach in a VR System which is based on the scenegraph system *OpenSG* [13] and the physics library *bullet* [19]. The software architecture is shown in Figure 3. The application is split into several components, while a main loop synchronizes the states between each of them. Since we use independent representations for the visible geometry and the geometry for the physics calculation, the results of the physics calculation need to be transferred into the scenegraph for updating the objects' poses. The generation of the physics proxies is explained in section 5.5. The internal capabilities of *OpenSG* are used to perform cluster rendering in a multi-projector environment.

For a general physics simulation, we use the *bullet* physics library, which, along with *PhysX* [21] and *Havok* [20], is a very popular physics engine for enabling physical effects in mainstream video games. Since its main purpose is geared toward video games, it especially allows for fast calculation results for many colliding rigid body objects and is well suited for real-time applications.

The user input is received by the tracking plugin, which runs in a separate thread to achieve the highest update rates possible. Our approach for a collision-based interaction is implemented as a plugin called "Physics interaction". At startup, the two plugins "Tracking" and "Physics interaction" perform a handshake procedure through an event handling mechanism. This makes it possible to use the current input data directly to calculate the pose of the physics hand, without going through the event system. The main communication between the components is event driven, such as state changes or collision events.

5.2 Tracking

We rely on an optical finger tracking system [18] to transfer the user's hand into the virtual environment. We use seven evenly spread cameras for a $3m^3$ volume. This system is capable of precisely determining the user's finger poses. The correct positions of the fingers' tips is especially important to allow precise manipulation. In Figure 4a, a finger tracking system is shown in conjunction with two virtual representations: tracking hand (middle) and physics hand (right). The tracking hand represents the poses we receive from our tracking system. In the image, the user moves his hand quickly in a horizontal direction from right to left to show the effects of the system's latencies. The physics hand model is paired with the tracking device's position values by fixed point connections (some kind of constraint in the physics simulation) at each phalanx as described below, but the physics simulation needs one frame to process that input. Thus, you see the position difference occurring between the different hand representations. The latency of the tracking system alone is around $20ms$ before the values are accessible in our system. These values are then used to calculate the physics hand pose, which is in terms the virtual hand representation used for actual interaction. The pose of the physics hand representation needs to be calculated by the physics engine since adjusting object poses directly within the physical simulation is not recommended. This would invalidate the internal solver state, what would lead to an unstable simulation.

The physics hand is almost at the same distance from the tracking hand as the tracking hand is to the user's hand. Therefore, we can conclude that in our setup, we roughly double the latency through our physics calculations. The total latency including the physics simulation is at least $60ms$ before the hand representation is rendered on screen–or two frames when running at 30Hz frame rate.

A robust coupling between user's hand (tracking hand respectively) and physics hand representation is very important. It directly controls the efficiency of the force transferred towards the virtual objects. To achieve the coupling, Borst et al. [1] used a spring-damper system. We found this approach to be quite limited, since laborious tuning of damping values was needed to achieve stable grasping results. The achieved coupling stiffness was also not sufficient to allow a constant force transfer to the grasped objects. The offset between real hand and physics hand becomes too large when fast movements were performed by the user. To overcome this limitation, we substituted this model by using fixed point constraints for coupling. In Figure 4b, the fixed point connection for a finger

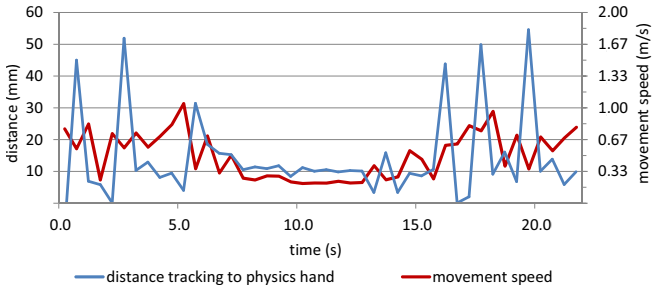

Figure 5: Distance between tracking data and physics hand over time while user performed an interaction. Values are measured from the index fingertip.

tip is visualized. The distance between blue and gray spheres is constrained to zero, enabling direct coupling. Being able to configure rigid bodies at three points on the tracking hand, the virtual hand model will then be able to follow the user's input quite directly and the correct orientation is implicitly conserved. The fixed point connections are applied as constraints to the physics library solver.

A typical interaction sequence is plotted in Figure 5. It shows the distance between tracking data and physics hand at its index fingertip during a common interaction task, due to latencies. Even though the movement speed is $1m/s$ at its maximum the offset of about $60mm$ is never exceeded. This is about one width of the palm of a hand, as shown in Figure 4a. These higher speeds are achieved during unconstrained movement of the hand. During the interaction with an object, the movement is slower, thus resulting in a shorter offset. Therefore we can show that a fixed connection allows a tight coupling to achieve optimal force transfer at high interaction speeds.

It must be noted that a fixed point connection yields the possibility to apply infinite input forces to the simulation system by moving the hand arbitrarily deep into virtual objects. Interestingly, we did not find this to be a problem in the proposed applications we have tested thus far. Deep penetration of the user hand result in a pop-through effect, which occurs due to a limited collision response. Nevertheless it is a problem that should be solved by a more sophisticated collision detection and response.

Duriez et al. [5] used a skeleton to apply their inputs to the virtual hand. This approach utilizes an accurate calibrated glove for each user's hand sizes. Since calibration is required for the best interaction results, we found that in real life the calibration step is often neglected or not desired at all. We therefore decided to transfer the user input to independent finger phalanxes; each finger consisting of three phalanxes. As we are separating the phalanxes, we do not apply any constraints between the finger parts. Through this, we allow different users with similar hand sizes to use the same calibration. Our hand model is robust enough to handle variations in finger lengths. We found that using two or three different calibration setups is sufficient to support most users, allowing them precise interaction.

Figures (6, 8, 10) clearly show the separated finger parts. The finger tip positions, as well as the joint positions between each part, are maintained through tracking input. The characteristics of our physics hand model will be discussed in the following section.

5.3 Hand Model

Based on the assumptions we made for coupling input data, we developed a hand model based on soft and rigid bodies to allow for robust, stable and versatile interaction. Considering that the fingers are utilized for precise interaction, we decided to simply use a rigid body only palm as in [1].

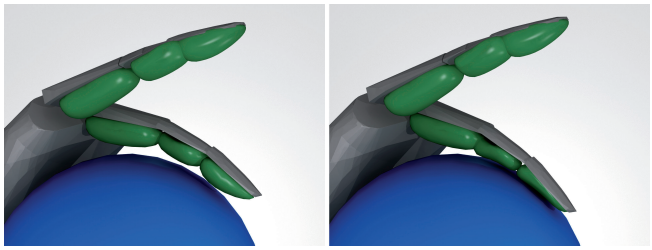

Figure 6: Developed hand model in lateral view while touching a virtual sphere. The deformation of finger-pads upon collision is clearly visible (right).

The phalanxes follow a different concept, consisting of two parts: a rigid body and a soft body. Figure 6 shows the virtual hand from a lateral perspective. Two fingers of the left hand are in touch with a virtual sphere. The twofold of rigid and soft parts stands out. The rigid parts acting as the skeleton are displayed in grayscale. They are coupled to the tracking input as previously explained. Each rigid finger part is accompanied by a deformable body, visualized in a green shade. The coupling between the two parts is achieved by fixing the penetrating particles' target positions to the movement of the rigid body. Each finger is made of three parts: the outer, middle and inner. The outermost part, representing the fingertip, is most important for precise interaction. The two other parts come in hand when a stable grasp is needed. They help to increase the contact area to a maximum and therefore apply a large amount of friction.

We use the soft finger-pads to simulate the increasing contact area when colliding, as simulated in Figure 6. Upon collision with an object, the bodies deform to match the shape of the obstacle and thus enable a realistic collision result by increasing the collision area. Another positive side effect is the smoothing of the force transfer towards virtual objects. The proposed fixed point coupling is characterized by a direct force transfer. The soft bodies have the effect of running this force transfer controlled. They act like shock absorbers which facilitates the interaction.

Each soft body consists of about 220 particles, depending on the phalanx size. For one hand with five fingers, we use 15 soft bodies (3275 particles total) and 16 rigid bodies. The extra rigid body represents the hand's palm, as previously noted. The palm serves as a collision partner on enclosing grasp gestures where a robust grasp is required, but in general no exact manipulation is performed.

We have tested our approach with up to four users' hands in complex scenes without detecting bottlenecks in stability or performance. The soft bodies' independence helps to maintain calculation speed, considering no further constraints have to be calculated except for the fixed point connection between tracking and the physics hand's rigid skeleton.

5.4 Collision Handling

For collision handling of the deformable finger-pads, we are using a penalty-based approach as described by Teschner et al. [17]. This collision handling approach was also used by Mueller et al. [11] in their initial shape matching algorithm and provides good results if collision forces are moderate. Penetration occurs if forces rise. The forces acting on scene objects are calculated and directly applied to the physics library. Although there is potential to further enhance the stability of grasping by better collision handling, we did not refine the provided model.

Self collision for the finger-pads is disabled as well as the collision between all phalanxes. The rigid bodies and soft bodies could therefore move freely, which helps in maintaining the hand pose directed by tracking input and reduces calculation costs. The colli-

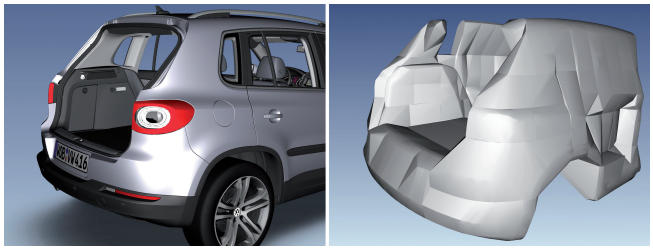

Figure 7: left: model of a car consisting of approx. 10 million polygons. right: convex decomposition of cars rear trunk, resulting in 132 convex objects.

sion between different hands is also disabled, including hands from different users. The collision detection is active both on the rigid as on the soft parts of the hand in conjunction with scene objects. Collision response is therefore only calculated between hand parts and scene objects, as well as between scene objects to allow for realistic object-object interactions.

5.5 Scene Preparation

Our goal is to support interaction with complex scenes. As we discovered, for most interaction tasks it is sufficient to approximate the visible geometry to receive an underlying, optimized physics scene. A physics calculation is then performed on a coarser model. For this, the visible geometry and simulation geometry are handled separately.

We are using a convex decomposition technique, developed by Lien [8], to extract our physics scene from visible geometry. Because of convex decomposition, the physics models can be efficiently handled by the physics library. By using a convex decomposition, we furthermore improve interaction speed, since collision detection does not need to be performed on a triangular basis. Objects are converted to a cluster of convex shapes, which are handled by the physics library.

A rough decomposition used for a car's boot trunk interaction is displayed in Figure 7. In this example, the car is roughly approximated by 132 convex objects, each consisting of 128 vertices. The vertex limit is proposed by the *bullet* library to efficiently calculate compound shapes.

In general, our scene models currently consist of up to 20 million triangles. By deriving a separate physics representation, we are able to perform all necessary calculations at interactive rates. This is applicable for interaction tasks where a plausible, but not precise, object reaction is needed. The accuracy of the approximation has to be considered for each application. Considering that in most grasping situations the user could not distinguish between a coarser and a precise approximation, the simulation result for the task itself depends directly on approximation quality. For assembly tasks an exact approximation is needed. Lien's method allows for approximation of different quality by taking volume error and concavity thresholds into account.

6 Dynamic Adapting of Stiffness

Different stiffness properties are needed to transfer the forces from grasping as directly as possible to the affected object while still maintaining a stable grasp. The latter could be done by increasing the contact area as stated in section 3. To extend the *FastLSM* through dynamic adapting stiffness, we need to precalculate several stiffness configurations which we can later select based on the actual requirements.

6.1 Precalculated Stiffness Regions

We precompute several stiffness configurations by using different region widths w. For each value of w, we store a neighborhood list R_s of the corresponding particles, as they represent different stiffness configurations of the pads, and therefore a smaller or larger neighborhood which influences a certain particle.

With just five different levels of effective stiffness we have achieved fast and stable grasping results. Consequently, we choose the number of region lists to be $L = 5$, and thus $w \in [1,5]$. Our approach is not limited to five different region widths. Considering the small amount of particles for each finger-pad and the stable interaction results, a larger variation in region widths was not necessary.

While it is important to be able to apply various stiffness setups, the selection of the most appropriate one which ideally fits each situation proves just as crucial. A dynamic adapting of stiffness is achieved by choosing a corresponding neighborhood list R_s, which is selected by the calculated index value w_s. Therefore, we introduce two thresholds which are used to influence the finger-pad's stiffness: forces and velocities. The thresholds are calculated independently for each body, depending on its collision state.

6.2 Velocity Threshold

The velocity threshold is used whenever the body is not in contact with other scene objects. This occurs when the user freely moves his hands around. Even in this case it is important to dynamically adapt the soft bodies' stiffness. It is important to hold a velocity-adapted configuration, especially for the change from free movement towards the case of collision. While slowly approaching an object, we want to get a different collision behavior than for fast movements. During the slow approaches, a soft configuration prevents uncontrolled force transfer, while a more rigid configuration provides a very direct force transfer in case of collision.

To calculate the velocity threshold, the rigid body parts of the hand model are used. The velocity is calculated for the center of mass from the rigid body that is directly connected to the soft body. The desired stiffness configuration is selected by taking a predefined maximum velocity into account.

We use a linear dependency to adapt the stiffness, clipping at a predefined maximum velocity value that is adapted to the movements' speeds that we measured in virtual environments–typically $1m/s$. The maximum velocity is divided by L to match the number of stiffness regions available. If the user moves his hands faster, a wider and therefore stiffer region configuration is selected. The resulting stiffness index w_s is then used for the next simulation step. It represents the index for selecting a region list. In our setup we use five precalculated region width configurations, therefore w_s is clipped to the interval $[1,5]$.

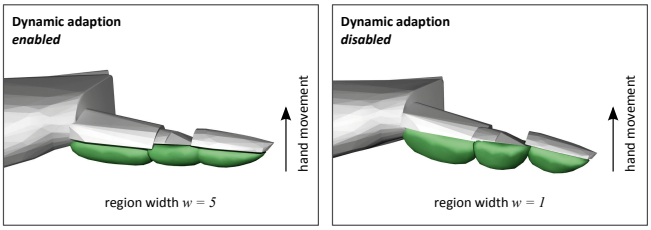

Figure 8: Hand is moved quickly in vertical direction. If dynamic adaption is disabled, the soft bodies deform through inertia.

By selecting an appropriate region width, we allow the finger-pads to follow the movement of the user. As shown in Figure 8, without the velocity threshold, the finger-pad experiences high deformations, considering that the fixed connections to the rigid body

indirectly result in a huge force which is applied to the body. With velocity threshold enabled, a larger region width is selected and thus the bodies' particles reach their target positions $\mathbf{g}_i$ (Equation 1) more rapidly.

6.3 Force Threshold

If a user closes his grasp, he wants a more stable object interaction. Therefore, a better force transfer towards the virtual object has to be guaranteed. This can be done by monitoring the acting forces. In case of contact, the desired stiffness is recalculated using force values at each particle position. For this, we use the distance between the current and target positions of each particle within a soft body. Forces acting on the finger-pads are calculated using their particles' velocities. Their velocity $\mathbf{v}_i$ in respect to the elapsed time is used to obtain the particles' unconstrained forces (Equation 4).

$$\mathbf{f}_i = m_i \mathbf{a}_i \quad \text{with} \quad \mathbf{a}_i = \frac{\Delta \mathbf{v}_i}{\Delta t} \tag{4}$$

With Listing 1, the necessary calculation steps are given to select an appropriate stiffness index for the next simulation step. The calculation is performed separately for each finger-pad.

Listing 1 Force threshold: calculate stiffness index w_s

```
1 for all finger-pad s do
2     f_avg  = 1/n Σ|f_i|                          // average force, (Equation 4)
3     f_norm = clip(f_avg, 0, f_max)               // normalize force by clipping
4     w_s    = round((L-1) f_norm/f_max) + 1       // stiffness index for part s
5 end for
```

At first the average force f_{avg} acting on all particles is calculated, where n is the number of particles in the finger-pad. The force is then normalized and divided by the number of available region widths L. A force f_{max} is defined beforehand, which represents the maximum force a user should apply to a body before the stiffest configuration is selected. Using the normalized force value, an appropriate index w_s is selected. In this case we use linear mapping. A non-linear function could be used to simulate the non-linear elasticity of the human skin. Since our results were satisfactory, we have not investigated further.

6.4 Algorithm Details

In each simulation step, the tracking data is used to update the constraints for the fixed-point coupling between the moving tracking hand and the rigid bodies of all hand parts. The tracking hand becomes a target configuration, while the algorithm in Listing 2 is performed to update the virtual hand model and scene objects.

Firstly, a single simulation step of the physics engine is performed. The fixed point constraints between tracking data and rigid parts of the hand representation are calculated by the solver of the *bullet* physics engine. It calculates the poses for all scene objects and provides collision forces that are used for our implementation of the *FastLSM*. As the rigid bodies' target positions are calculated, the target position for the fixed particles at the finger-pads is applied. The forces that were acting on the scene objects were applied to the configuration of the virtual hands. For each finger-pad, its deformation state is calculated. Therefore, in line 5 of Listing 2, the current stiffness index w_s is used to select a region list R_s, which is used as the region list R_i in Equation 1 to calculate the target positions $\mathbf{g}_i$ of the finger-pads' particles.

For each pad, penalty forces are calculated using the approach of Teschner et al. [17]. All friction forces between finger-pads and scene objects are applied as penalty forces to the scene objects.

After calculating the position of all scene objects, the velocity and acting force on the rigid body of each phalanx is calculated by

Listing 2 Simulation step

```
1  perform bullet stepSimulation()
2  apply forces to extended FastLSM
3  for all finger-pads s do
4      for all particles i do
5          select R_s as R_i using w_s
6          calculate g_i (Equation 1)
7          calculate v_i(t+h) (Equation 2)
8          calculate x_i(t+h) (Equation 3)
9      end for
10     perform collision detection [17]
11     apply collision forces to bullet
12     if s is in contact then
13         calculate w_s using Listing 1 lines (2-4)
14     else
15         calculate w_s using velocities
16     end if
17 end for
```

the rules explained in section 6.2 and section 6.3. When performing this for each finger-pad, an appropriate stiffness value w_s for the next simulation step is selected.

At the end of each simulation step, the pose of the visible representation is synchronized with the calculated pose of the physical representation. The physics geometry is generally not visualized, but may be helpful for debugging purposes. To reduce the risk of a bottleneck, we use mutual exclusion to synchronize the asynchronous threads running the rendering and physics calculations. Because the finger-pads consist of a small number of triangles, we update each vertex position by trilinear interpolation of the surrounding particles' positions on the CPU. Rivers et al. [14] also presented a way to accelerate soft body rendering through a GPU shader.

7 Simulation Results

In Figure 9, simulation values of a short interaction sequence are given. The values represent the movement of an index fingertip. They were collected at key frames in a $0.5s$ interval. The figure shows the ratio between velocities, acting forces, the selected region width representing the pad's stiffness and calculation time for the physics simulation. The benchmark was performed with a simple scene, consisting of few objects (see Figure 10 and 11) on a test system running Intel Core i7-940. Two user hands were enabled.

The figure is split into four plots. Whenever the force is at zero, the user moves his hand freely around. In these cases the stiffness index is solely selected based on velocities. The change in velocity directly adapts the region's width of the finger-pad.

Whenever the force is greater than zero, the user directly interacts with a scene object. When the force gets smaller by the eighth second, the user loosens his grasp. This results in selecting a smaller stiffness index.

In this representative benchmark, the stiffness is often at higher values, indicating an experienced user who does not hesitate to move or to grasp objects in a resolute way. This is also reflected by the relatively fast movement speeds throughout the interaction sequence. The high stiffness ensures a quite direct force transfer, enabling stable grasps.

The calculation time is measured for the whole simulation step, not just the simulation time for the fingertip. Nonetheless, it never exceeds $10ms$ throughout the 25 seconds of interaction. You may notice that the simulation time is quite constant, regardless of whether or not the user interacts. This ensures a smooth interaction experience.

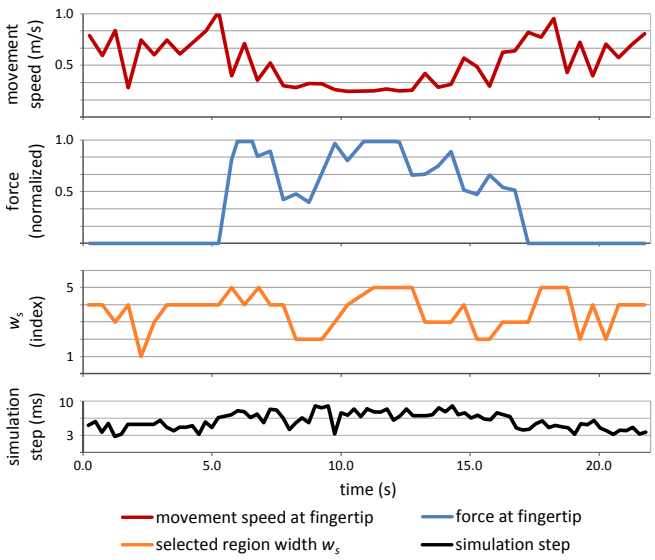

Figure 9: Interaction over time, showing the ratio between calculation times, velocities and selected region width.

In section 5.5 we mentioned that our scene mainly consists of compound convex objects. Through this, we are able to simulate a full-featured car interior with up to 30 frames/s, but with a loss in accuracy as the trade-off. Physics simulation with two hands is done in about 1 to 10ms depending on collision state and stiffness value. Propagating scene changes to the display system and rendering itself takes another 33ms. With this result, we always maintain interactive frame rates, all the while the latency does not seem to be too disturbing for the users while performing their actual task.

We did not perform an extensive user study based on our hand-model as generally only selected experts are using the virtual grasping. The following findings therefore reflect discussions and comments made by several users who interacted with the system without prior instructions. The users did not get any additional visual or other feedback upon collision besides that the virtual objects just start moving when they were touched.

Some users, who never used collision-based grasping before, were able to interact quite well. After a short time of usage, most users were quite eager in performing very precise interactions by stacking or combining virtual objects in the scene. As observed in other grasping approaches [1], we also had a wide variation among users at their first grasping attempts. The users mentioned that their learning curve was rampant, as they realized to grasp as in the real world rather than closing their hands to form a fist. The users liked that the objects behave realistically, which helps to manipulate objects quite fluently. The system latencies did not seem to influence the interaction results that much, as no one mentioned that the system reacts too slowly.

Some users noticed that the interaction behavior was different, depending on how fast they reached towards the objects. They stated that they saw quite well how the virtual object reacts based on their input. For most users, the movement of the object served as a sufficient clue indicating that their grasp was successful. Disabling the soft pads and thus relying on the rigid parts of our virtual hand, it became evident that we needed much higher friction values to establish stable grasps. As a result, it was harder to unhand objects and not possible to glide over objects without applying huge friction forces.

There were no instances in which the users claimed that the visual world does not parallel the interaction result. They were somewhat surprised to hear about the optimization achieved by using convex decomposition and thus not acting on the visible geometry itself.

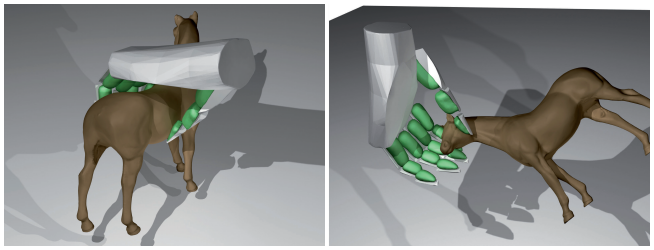

Figure 10: Unconstrained interaction with a horse model. The finger-pads adapt to the geometries' shape, enabling stable and robust interaction.

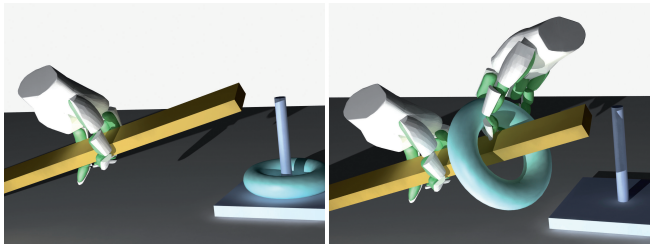

Figure 11: Two handed interaction with non-constrained objects. Collision response between torus and stick is enabled through physics simulation.

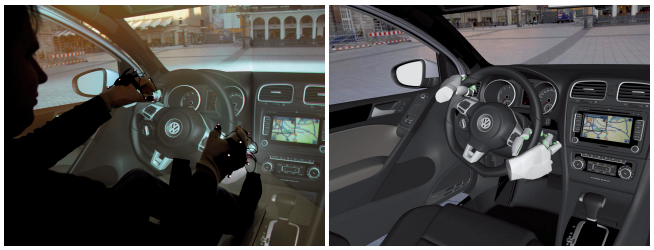

Figure 12: Interaction within an immersive display system. A user interacts with a constrained steering wheel using both hands, thus reproducing a real-world interaction.

Overall, the users' findings indicate that the virtual scene provides very solid feedback regarding object manipulation, enabling very precise manipulation tasks. The sensitive manipulation with the fingertips especially benefits from the finger-pads as shown in Figure 10. One thing the users complained about was the unnatural appearance of the virtual hand, since it consisted of independent disconnected finger phalanxes.

Most users that were utilizing our virtual grasping tended to use just one hand for interaction. When given more complex tasks and asked to use two hands, the users did not find it any different from using just one hand. They were also able to perform tasks that included object-object interaction as well. In Figure 11, a two handed interaction is shown. The task is to put a torus on a stick. This kind of task could be easily fulfilled using our approach–even for novice users.

Interaction with constrained objects is just as possible as with freely movable objects. In a more complex scene, the users were able to use typical constructional elements that were constrained. Figure 12 shows a user interacting with a virtual car. As a matter of course, he uses both hands to control the steering wheel. The

wheel is constrained using a revolute joint, thus allowing for rotation around one axis. The interaction is very similar to how it would be in real life, in the sense of being able to rotate the wheel simply based on friction by putting the hand on top of the steering wheel.

8 Conclusions and Future Work

Our new hand model is based on soft bodies coupled to a rigid body hand skeleton. It allows for very precise and robust finger-based grasping, manipulation and releasing of virtual objects. The dynamic adaptation of the stiffness values of the soft bodies handled by the extended *FastLSM* algorithm is the basis for the robustness of our approach. An important advantage is the implicit friction model which results from the pressure-based increasing and decreasing of the contact area of the simulated finger phalanxes. In contrast to pure rigid body-based interactions, where only a single friction value per object can be selected, the dynamic stiffness adaptation results in a stable behavior for a wide variety of parameters. Our implementation indicates that the system runs in real-time for complex scenarios.

There are various ways to further improve the realism and the scope of our physically-based grasping approach.

- Currently, only the finger phalanxes are represented by a soft body. Enclosing grasps involve the palm and thus the palm should also be represented by a single large or multiple smaller soft bodies. Care must be taken in choosing the shape of the soft body for the palm, since it plays an important role in achieving stable behavior.

- In extreme cases, the absolute stiffness of the finger-pads can be insufficient. Powerful grasps might result in a collapsing finger-pad and thus the grasping occurs through the rigid body hand skeleton. Since the finger-pads are exchangeable on the fly, it might be possible to choose a different deformation algorithm depending on the situation. Approaches that specialize in simulating very stiff objects [4, 3] could have an advantage over the *FastLSM* algorithm for larger forces.

- A skinned hand representation would significantly improve the visual appearance of our hand model. One solution might be to use locally different stiffness regions per soft body as described in [16]. However, further investigations on the achievable absolute stiffness as well as how to couple such a skinned hand in an optimal way to the tracked real hand remain necessary.

The combination of a rigid body skeleton and a set of soft bodies with different and adaptive stiffness values could also be a promising approach for a physically-based representation of an entire human. Our soft hand model is a first step in this direction.

Acknowledgements

We would like to thank the team of the Volkswagen VRlab for their support with the hardware setup and the enriching discussions, the Large Geometric Models Archive for the models we've used for testing our approach, and the reviewers for their detailed and constructive comments.

References

[1] C. W. Borst and A. P. Indugula. Realistic virtual grasping. In *Virtual Reality Conference (VR), 2005 IEEE*, pages 91–98, 320, 2005.

[2] M. Ciocarlie, C. Lackner, and P. Allen. Soft finger model with adaptive contact geometry for grasping and manipulation tasks. In *Proceedings of the Second Joint EuroHaptics Conference and Symposium on Haptic Interfaces for Virtual Environment and Teleoperator Systems*, pages 219–224, Washington, DC, USA, 2007. IEEE Computer Society.

[3] R. Diziol, D. Bayer, and J. Bender. Simulating almost incompressible deformable objects. In *Virtual Reality Interactions and Physical Simulations (VRIPhys)*, Karlsruhe (Germany), Nov. 2009.

[4] R. Diziol, J. Bender, and D. Bayer. Volume conserving simulation of deformable bodies. In *Proceedings of Eurographics*, Munich (Germany), Mar. 2009.

[5] C. Duriez, H. Courtecuisse, J. P. de la Plata Alcalde, and P.-J. Bensoussan. Contact skinning. In *Eurographics 2008 (short paper)*, pages 313–320, New York, NY, USA, 2008.

[6] B. Froehlich, H. Tramberend, A. Beers, M. Agrawala, and D. Baraff. Physically-based manipulation on the responsive workbench. In *Virtual Reality Conference (VR), 2000 IEEE*, pages 5–11, 2000.

[7] 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*, 13:732–747, July 2007.

[8] J.-M. Lien. *Approximate Convex Decomposition and its Applications*. PhD thesis, Texas A&M University, 2006.

[9] M. Moehring and B. Froehlich. Pseudo-physical interaction in a virtual car interior. In *Proceedings of IPT/EGVE Workshop*, pages 181–189, 2005.

[10] M. Moehring and B. Froehlich. Enabling functional validation of virtual cars through natural interaction metaphors. In *Virtual Reality Conference (VR), 2010 IEEE*, pages 27–34, 2010.

[11] M. Mueller, B. Heidelberger, M. Teschner, and M. Gross. Meshless deformations based on shape matching. *ACM Trans. Graph.*, 24(3):471–478, 2005.

[12] M. Ohta, Y. Kanamori, and T. Nishita. Deformation and fracturing using adaptive shape matching with stiffness adjustment. *Computer Animation and Virtual Worlds*, 20:365–373, June 2009.

[13] D. Reiners. *OpenSG*. PhD thesis, TU Darmstadt, Darmstadt, September 2002.

[14] A. R. Rivers and D. L. James. Fast LSM: Fast lattice shape matching for robust real-time deformation. In *SIGGRAPH '07: ACM SIGGRAPH 2007 papers*, pages 82–87, New York, NY, USA, 2007. ACM.

[15] H. Salzmann, J. Jacobs, and B. Froehlich. Collaborative interaction in co-located two-user scenarios. In *Joint Virtual Reality Conference (JVRC) - the 15th Eurographics Symposium on Virtual Environments, ICAT, EuroVR*, pages 85–92, Lyon, France, 2009.

[16] D. Steinemann, M. A. Otaduy, and M. Gross. Fast adaptive shape matching deformations. In *SCA '08: Proceedings of the 2008 ACM SIGGRAPH/Eurographics Symposium on Computer Animation*, pages 87–94, Aire-la-Ville, Switzerland, 2008. Eurographics Association.

[17] M. Teschner, B. Heidelberger, M. Mueller, D. Pomeranets, and M. Gross. Optimized spatial hashing for collision detection of deformable objects. In *Proceedings of VMV'03*, pages 47–54, 2003.

[18] Website. *Advanced Realtime Tracking GmbH*, Dec. 2010. `http://www.ar-tracking.de`.

[19] Website. *Bullet Physics Library*, Dec. 2010. `http://bulletphysics.org`.

[20] Website. *Havok Physics*, Dec. 2010. `http://www.havok.com`.

[21] Website. *NVIDIA PhysX*, Dec. 2010. `http://developer.nvidia.com/object/physx.html`.

[22] G. Zachmann and A. Rettig. Natural and robust interaction in virtual assembly simulation. In *Eighth ISPE International Conference on Concurrent Engineering: Research and Applications (ISPE/CE2001)*, volume 1, pages 425–434, West Coast Anaheim Hotel, July 2001.