A Generalized God-Object Method for Plausible Finger-Based Interactions in Virtual Environments

Jan Jacobs*
Group Research Virtual Technologies
Volkswagen AG

Michael Stengel†
Computer Graphics Lab
TU Braunschweig

Bernd Froehlich‡
Virtual Reality Systems Group
Bauhaus-Universität Weimar

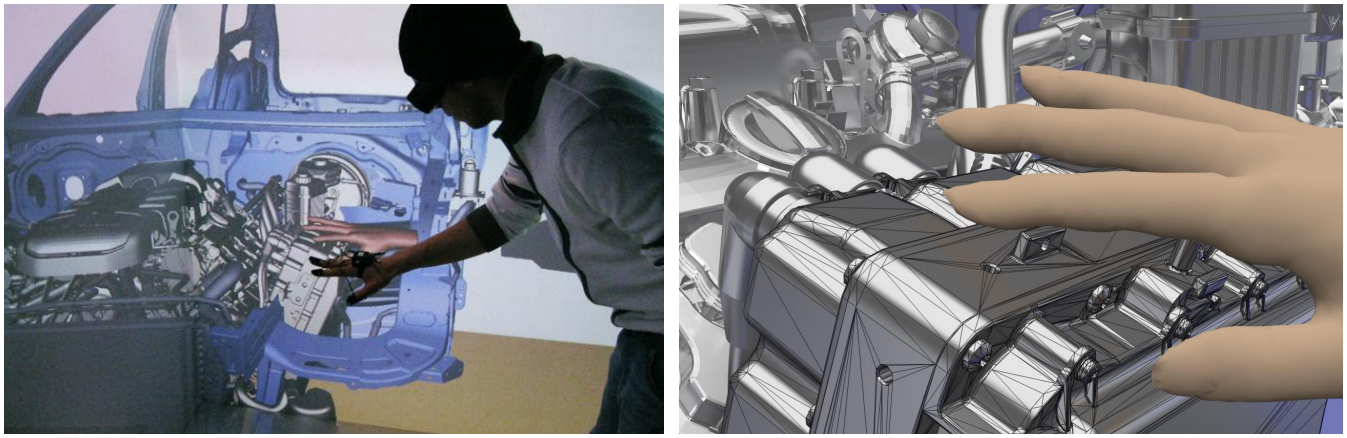

Figure 1: The God hand: The fingers of the real hand may penetrate an object (left) while the representation of the virtual hand remains in a plausible position outside of the object.

ABSTRACT

We generalize the six degree-of-freedom God-object approach to enable its use for multi-finger interactions in virtual environments. The connected finger phalanxes are modeled as multiple constrained God objects. The mutual interdependencies between multiple God objects are resolved using Gauss' principle of least constraint. This generalization of the God-object method allows us to avoid the penetration of multiple fingers and their phalanxes with objects within a physically simulated virtual world.

Our observations indicate that the generalized God-object approach leads to plausible collision-free positions and motions of the phalanxes of the user's fingers during complex six degree-of-freedom manipulations, while artifacts such as artificial friction or a stuck hand are avoided.

Index Terms: I.3.6 [Computer Graphics]: Methodology and Techniques—Interaction techniques; I.3.7 [Computer Graphics]: Three-Dimensional Graphics and Realism—Virtual reality; G.1.6 [Mathematics of Computing]: Numerical Analysis—Optimization; H.5.2 [Information Interfaces and Presentation]: User Interfaces—direct manipulation

1 INTRODUCTION

Most real-time physics simulation engines are optimized for handling reasonably large virtual models in a mostly plausible way. An often encountered trade-off is the limited treatment of large collision forces, and as a result, physically impossible deep interpenetrations of objects and pop-through effects may occur. However, users interacting with a physically simulated virtual world may unintentionally create such large collision forces and will perceive penetrations of their visual body representation with the manipulated geometry. While this is not only disturbing, it also leads to a noncredible interaction behavior. Although the God-object approach by Ortega et al. [28] is currently the standard for efficiently computing penetration-free interactions of two virtual objects (e.g. between the virtual representation of a pen in the user's hand and a virtual object in the scene), it is not suitable for finger-based manipulation.

We introduce a generalization of the six degree-of-freedom (DOF) God-object approach [28] to support multiple God objects for enabling a penetration-free interaction with a multibody simulation of dynamic objects. Our work is based on deriving the mathematical and algorithmic foundations to resolve the mutual interdependencies among multiple interacting God objects. We solve the dynamics equations of the corresponding N-body system and its inherent constraints using Gauss' principle of least constraint [15]. Based on the resulting accelerations of all bodies and a continuous collision detection, we successfully inhibit undesirable interpenetrations among the multiple God objects as well as among God objects and objects in the virtual environment. For simulating finger-based interaction with a physically simulated virtual world, we developed the God-hand model (Figure 1). Each finger phalanx is represented as an individual God object. The phalanxes of each finger are tied together and limited in their local movement by constraints. For enabling the simulation of complex scenarios, we couple the accurate simulation of God objects with a less accurate but faster physics simulation engine, which handles the interaction among the remaining objects in the virtual environment.

Assembly simulations and ergonomic studies in the automotive industry are often performed using finger-based interactions with-

*e-mail:jan.jacobs@volkswagen.de
†e-mail:stengel@cg.cs.tu-bs.de
‡e-mail:bernd.froehlich@uni-weimar.de

IEEE Symposium on 3D User Interfaces 2012
4-5 March, Orange County, CA, USA
978-1-4673-1205-9/12/$31.00 ©2012 IEEE

out haptic feedback [23, 17]. In these scenarios penetrations of the user's fingers with the virtual objects occur quite often. However, Burns et al. [10] emphasized that our proprioception reacts much less sensitively to incorrect or changing real-virtual movement mappings than our visual system reacts to interpenetrations of virtual body representations and virtual objects. Consequently, these visual interpenetrations should be avoided by always keeping the virtual finger representations in a plausible position outside of a virtual object.

The main contribution of our work is a generalized 6-DOF God-object method for a precise simulation of the interaction among multiple God objects. We verified our approach by developing and implementing a God-hand model for finger-based interaction with a physically simulated environment. Our observations reveal that our system always generates penetration-free and plausible virtual finger positions and finger motions even if the real hand deeply penetrates a virtual object. Furthermore, common physical simulation artifacts such as artificial friction or finger phalanxes that are getting stuck inside an object do not occur. Initial experiments with the coupling of our God-object simulation with a less precise, penalty force-based real-time physics simulator show that our God-hand representations smoothly interact with complex virtual models.

2 Related Work

Recently realistic finger-based grasping of virtual objects has become an important research topic. Weber et al. [30] showed a semi-automated grasping approach, where a virtual hand automatically executes an ideal grasping gesture once the user's hand comes close to an object. This approach leads to stable, non-penetrative grasp gestures but lacks the flexibility to consider the movements of the user's fingers. Holz et al. [16] developed a grasping approach that uses friction cones to establish a grasp between finger pairs and objects. This approach was extended by Moehring et al. [22] to support multiple hands and the grasping stability was increased by the introduction of grasping proxies. While these approaches attempt to mimic physical behavior, they cannot properly deal with finger-object interpenetrations and neglect object-object interactions.

Bergamasco et al. [6] introduced the use of collision-based grasping and point-based force calculations to achieve physically plausible object behavior. Borst et al. [8] pioneered the use of physics engines for the precise simulation of finger-based grasping in virtual environments. Jacobs et al. [17] improved the grasping stability of this approach by introducing an advanced hand model consisting of a rigid-bone skeleton and soft bodies for representing the contact surfaces of the fingers. Both approaches use real-time physics engines which are optimized for a plausible simulation and may allow for object interpenetrations in complex situations. A more reliable collision handling is needed to address the needs of complex virtual assembly simulations.

According to Erleben [14], a dynamics simulator consists of two core components: collision detection and simulation. The collision detection is responsible for the quality of the generated contact points between colliding objects. Precise computation of contact points is ideally performed with a continuous collision detection. Kockara et al. [18] provided an overview of commonly used collision detection approaches.

Baraff [2] developed an iterative approach for the simulation of contact forces. The computation of constraint forces between two dynamic bodies requires solving a linear complementarity problem. This approach results in a precise simulation of object movements whereby deep mutual penetrations are avoided. Redon et al. [29] demonstrated that Gauss' principle of least constraint [15] leads to a similarly efficient computation of object movements as compared to Baraff's approach [2], even though it is a quadratic programming problem. However, Redon's approach is better conditioned if complex objects or many contact points are involved.

Bender [5] used an iterative approach for resolving contact forces and constraints in an impulse-based dynamics simulation. The number of required iterations for precise results varies depending on the complexity of the situation, particularly if penetrations have to be resolved. Alternatives to impulse-based simulations are position-based approaches [26, 25] or mass-spring systems for deformable objects [27]. The latter approaches mostly focus on plausible visual effects rather than on a correct or penetration-free simulation.

There are a number of different approaches for dealing with collisions. A simple approach computes penalty forces if the physical state is incorrect. These forces are supposed to move the objects back into a valid physical state [24]. This penalty force-based approach is computationally expensive if high precision is needed. It also requires the use of implicit solvers if the system gets stiff. Object penetrations can also be resolved by the iterative use of impulses [21, 4], which may require a large number of iterations since each iteration may introduce new collisions. An alternative is the use of compensation forces, which are computed by solving a linear complementarity problem [1]. This approach simultaneously considers all contact points and computes a precise and penetration-free state in a single step. However, an efficient implementation is rather involved such that the other approaches based on penalty forces or impulses are more often used.

Haptic feedback requires very high update rates to provide continuous collision feedback to the user. In this context, a single contact object serves as a proxy for the haptic device. The God-object method for 3-DOF haptic devices [35] and its extended 6-DOF version [28] generate a penetration-free position for the device proxy during haptic interaction with rigid bodies. The God object (device proxy) always remains at a plausible position on the surface of an object, even though the haptic device may actually penetrate the object. Both methods work at sufficiently high update rates for the haptic devices as well as for the visual display of the device proxy. However, mutually influencing dynamic bodies were not considered.

Zachmann et al. [33] used virtual proxies for generating pseudo-physical object motions during two-handed manipulations. Their approach generates a ghost object for a tracked human hand, which is always kept close but outside of a manipulated object using a minimization approach. Borst et al. [8] developed a spring model for the whole hand, which avoids hand-object penetrations. This spring-damping approach is limited in the amount of force that can be transferred, which in turn is limiting in the interaction with large objects and for two-handed interactions.

3 Six Degree-of-Freedom God-Object Method

To introduce the extended God-object method, we first provide an overview of the regular 6-DOF God-object method by Ortega et al. [28]. They have shown how Gauss' Least Constraint Principle can be used for a penetration-free 6-DOF simulation of dynamic objects. Their approach targets highly accurate haptic rendering of a virtual scene.

The simulation setup by Ortega et al. contains an interactive rigid body, which is movable by the user with the haptic device. Furthermore, the scene contains a static rigid body, which can be "touched" but not moved by the user. The algorithm calculates a visual representation of the interactive object and a force, which can be used by the haptic device for haptic rendering.

For haptic interaction, it is important that the user is not able to pierce an obstacle body (pop-through). Therefore, the 6-DOF God-object approach considers two representations of the interactive body, *target* and *God object*. At all times, the target is located at the position of the haptic device. The target is invisible to the user, whereas the God object is visualized. It is connected to the target by constraints. However, the God object remains outside of the volume

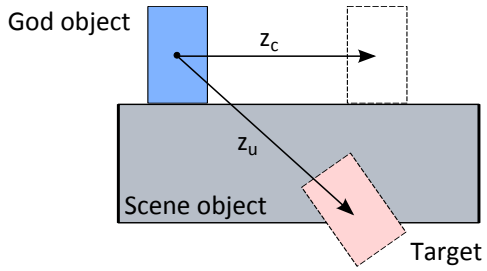

Figure 2: Relationship between God object and target for the time of penetration.

of the other rigid body even if the target penetrates the body caused by the position of the haptic device. The relationship between both bodies is shown in Figure 2. Besides a very natural interpretation of the force action, the penetration-free visualization of the contacting bodies is a valuable side-effect for the observer. The pose of the God object is guided by using a constraint, which establishes a relationship between target and God object. If the target resides in a contact-free condition, the God object is also brought into this condition by the constraint, which is ideal for this case. If the target is moved by the user into a penetration state with the obstacle body in the scene, contact points are generated. Using these contacts, a pose on the surface of the obstacle is calculated, in which the least resistance acts on the God object. Hence, the simulation is formulated as an optimization problem.

For their approach, Ortega et al. assumed continuous collision detection (CCD). This completely avoids missing collisions and allows avoidance of perceptual penetrations. For computing such a state, an acceleration vector for the God object is estimated by using Gauss' principle of least constraint. This principle can be postulated by minimizing the following Equation 1.

$$G(a) = \frac{1}{2}\|a - a_u\|_M^2 \tag{1}$$

Vector a_u describes the acceleration of a rigid body assuming unconstrained movements. The body is specified by the 6×6 mass matrix M. The principle of least constraint can be understood in a way that it chooses one specific acceleration a from the set of possible accelerations, which minimizes Equation 1. This acceleration a takes into account all of the external constraints on the body and matches the unconstrained acceleration as close as possible. Therefore, the acceleration is in a sense *minimal* for all accelerations and actually taken for moving the rigid body.

As previously mentioned, in the God-object approach a connection between target and God object is simulated. This is done by the virtual coupling equation (Eq. 2). Additionally, to avoid artifacts in force feedback, Ortega et al. formulate a constraint-based coupling.

$$z_u = k_s(x^{Target} - x^{God}) \tag{2}$$

In Equation 2 vector z_u is a stacked 6D vector containing respective linear and angular accelerations of the God object. The poses of both considered bodies, each consisting of position and orientation, are also specified by 6D state vectors, x^{Target} and x^{God}. The variable k_s is a coupling constant and defines the tightness of the connection of the target and God object.

Considering all constraints acting on the God object, the constrained acceleration z_c can be estimated by the following minimization (Eq. 3). Ortega et al. only took contacts into account for constraining the God object. All constraints are stacked in a Jacobian matrix $\mathbf{J}_{n\times 6}$ for n contacts.

$$z_c = \arg\min\left\{\frac{1}{2}\|z - z_u\|_M^2 : \mathbf{J}z \geq 0\right\} \tag{3}$$

This minimization system was used by Ortega et al. to estimate the acceleration of a single God object [28]. They have shown that this approach is efficient for calculating penetration-free contact states and is also inexpensive with respect to computation time.

4 Simulating Multiple God Objects

A major challenge for the simulation of multiple God objects is the proper consideration of potential reciprocal force transmissions among them. We solve this problem by calculating valid poses for multiple God objects in an N-body simulation. The following steps constitute the main components of our approach:

1. Computing of inner constraints and contact constraints based on accelerations for all dynamic bodies.
2. Stacking constraints within a Jacobian matrix.
3. Deriving dynamics equations from the Jacobian matrix considering Gauss' Principle of Least Constraint.
4. Calculating a generalized acceleration vector z_c by finding an optimal solution for the dynamics equations.
5. Computing the constrained target positions using accelerations from the generalized acceleration vector a in an Euler integration.
6. Computing the minimal time-of-contact (TOC) using continuous collision detection among all bodies using their last and new positions.
7. Using minimal TOC for time step adaptation during Euler integration for a penetration-free update of the targets' body positions.

In the following section 4.1, we will show how to generalize a constraint solver, which is necessary for (3) to be able to perform the calculations. In Section 4.2 we discuss how a set of constraints can be defined for multiple bodies. The basic handling of the TOC information is not explained before Section 6.3 as the implementation details need to first be introduced. In this section we will also explain the integration scheme used with our implementation.

4.1 Least Constraint Solver

The most important aspect is the correct treatment of body contacts. Physical contacts have to be simulated whenever the collision detection generates a contact point of two bodies A and B. The contact point is determined by a contact point coordinate r and a contact normal n which, by definition, always points from body B towards body A. Depending on the movement of the body, there are different types of contact. To determine the contact type, the relative velocity v_{rel} at the contact point along the contact normal can be calculated by

$$v_{rel} = n\,(\,\dot{r}^B - \dot{r}^A\,). \tag{4}$$

The difference of velocities at the contact point with regard to body A and B is defined in world frame XYZ. The velocities are calculated by taking the first derivatives of the body points at the contact, $\dot{r}^A$ and $\dot{r}^B$. Figure 3 explains the relation of a point on the body and the world frame. Physical contacts should always be determined by using a CCD for the God objects, since this approach avoids missing collisions. Once physical contacts are determined, they must be correctly treated such that objects do not penetrate. In our method we calculate compensation forces in a way that they always apply $\dot{v}_{rel} \geq 0$. To achieve a simulation with an arbitrary

number of objects, Equation 3 needs to be generalized. For this purpose, the mass matrix M is replaced by the generalized mass matrix $\mathbf{M}$ of all dynamic bodies. Furthermore, the Jacobian matrix now contains constraints of all bodies in the scene. This leads to the following formulation of the minimization problem for N rigid bodies:

$$z_c = \arg\min \left\{ \frac{1}{2} \|z - z_u\|^2_{\mathbf{M}} : \mathbf{J}z \geq c \right\}. \tag{5}$$

The vector $c \in \mathbb{R}^{1\times m}$ at the right-hand side may deliver non-zero values for m general acceleration-based constraints. The introduction of this vector c is necessary to allow for the handling of joints. In this formulation the generalized acceleration vectors z and z_u are used, which contain the acceleration components of all considered bodies.

The minimization equation is put into the form of a projection problem and solved as such. This is in analogy to the method of Ortega et al. [28]. In the first step, the generalized mass matrix $\mathbf{M} \in \mathbb{R}^{6N\times 6N}$ and the Jacobian matrix $\mathbf{J} \in \mathbb{R}^{m\times 6N}$, containing the constraint equations for all bodies, are determined. According to Redon et al. [29], it is additionally necessary to compute a factorization of the generalized mass matrix. This is always possible, since the mass matrix for a rigid body is symmetric and positive-definite. A factorization can be achieved numerically through a QR-decomposition [12].

The factorization of $\mathbf{M}$ results in an upper triangular matrix $\mathbf{Q} \in \mathbb{R}^{6N\times 6N}$, for which Equation 6 holds.

$$\mathbf{M} = \mathbf{Q}^T\mathbf{Q} \tag{6}$$

The constrained acceleration can not directly be solved in the form of Equation 5, because the constraints are formulated by the Jacobian matrix in the secondary condition.

Through the Equation 7, a solution vector $k \in \mathbb{R}^{1\times 6N}$ needs to be calculated.

$$\mathbf{J}k = c \tag{7}$$

With this vector, the factorization matrix $\mathbf{Q}$ and the unconstrained acceleration $z_u \in \mathbb{R}^{1\times 6N}$ the vector $s \in \mathbb{R}^{1\times 6N}$ is determined.

$$s = \mathbf{Q}(k - z_u) \tag{8}$$

After calculating the inverse $\mathbf{Q}^{-1}$, the matrix $\mathbf{J}_Q$ can be calculated by

$$\mathbf{J}_Q = \mathbf{J}\mathbf{Q}^{-1}. \tag{9}$$

With this matrix $\mathbf{J}_Q$ and the vector s it is possible to formulate the following minimization equation.

$$\lambda = \arg\min \left\{ \left\| \frac{1}{2}\mathbf{J}_Q^T\lambda - s \right\|^2 : \lambda \geq 0 \right\}. \tag{10}$$

The solution of this equation is a *least squares problem* with the condition of a non-negative solution (*non-negative least squares*, NNLS). For this standard problem, efficient numerical solvers exist [7, 9]. It should be emphasized that the required matrix for this minimization is always sparse. The same applies to the matrix $\mathbf{J}$ in the solution of the equation system according to k. This fact should be considered for selecting an appropriate and efficient numerical solver.

The solution λ provides the constrained acceleration $z_c \in \mathbb{R}^{1\times 6N}$ via the final equation

$$z_c = \mathbf{Q}^{-1}\mathbf{J}_Q^T\lambda + z_u. \tag{11}$$

This generalized acceleration vector allows for the movement of all components of a scene. The definition of constraints is explained in the following section.

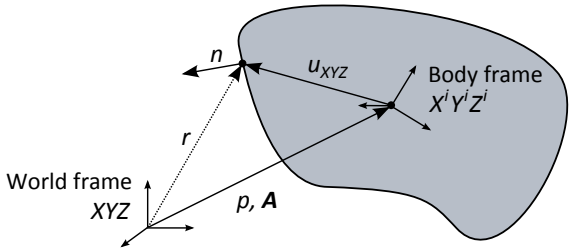

Figure 3: Definition of reference frames.

4.2 Constraints

Constraints are necessary to correctly deal with relationships between objects. For this purpose, constraint equations are introduced. Constraints usually act on a single or between two different bodies. For clarifying the notation, world frame and body frame are described in Figure 3. The world frame is defined as XYZ . The pose of the local reference frame $X^iY^iZ^i$ for a body i is defined by its center of gravity p and the rotational matrix $\mathbf{A}$, resulting in a 6D state vector x^i of the body. Any point within the body is noted as r, whereas u denotes the distance between r and p. This distance is u_{XYZ} in world frame.

There are two types of constraints to restrict the movement of bodies: non-holonomic and holonomic.

Holonomic constraints

Holonomic constraints always reduce the allowed degrees of freedom. This includes various constraints such as prismatic joints, revolute joints or spherical joints.

Holonomic constraints between two bodies that are not dependent on time can be described by the Equation

$$C(x^i, x^j) = 0. \tag{12}$$

In this equation, both bodies are defined by their 6D state vectors. The constraint equation between two bodies is considered to be satisfied if it provides the zero vector for the given parameters.

Non-holonomic constraints

Non-holonomic constraints can not be solved by Equation 12, but are needed to define permissible ranges of movement, such as for joint limits. In general, a non-holonomic constraint can be formulated by the following inequality

$$C(x) > 0. \tag{13}$$

The degrees of freedom remain intact within the defined permissible range.

For modeling a constraint between two bodies B^i and B^j, their respective states have to be considered.

$$C(x^i, x^j) = C(p^i, \mathbf{A}^i, p^j, \mathbf{A}^j) = 0 \tag{14}$$

In this formulation the position constraint depends on the position vectors p and the orientation matrices $\mathbf{A}$ of both bodies. The constrained movement can now be converted into an acceleration-based constraint by the second derivative $\ddot{C}$.

$$\dot{C} = \frac{dC}{dt} = \frac{\partial C}{\partial x}\dot{x}^i + \frac{\partial C}{\partial x}\dot{x}^j = J^i y^i + J^j y^j \tag{15}$$

$$\ddot{C} = J^i\dot{y}^i + \dot{J}^i y^i + J^j\dot{y}^j + \dot{J}^j y^j \tag{16}$$

In Equation 15, y^i and y^j describe the rate of changes of positions and orientations, which can be called generalized velocities.

In generalized coordinates, the acceleration-based constraint equation has the form

$$\ddot{C}(z^i, z^j) = \ddot{C}(a^i, \alpha^i, a^j, \alpha^j) = 0, \tag{17}$$

whereas z^i and z^j describe the generalized acceleration vectors containing linear and angular accelerations, a and α. The God-object method considers the simulated bodies as quasi-static. This means, that a speed of zero is assumed at each time step. Using this assumption, we can simplify the constraint equation for two involved bodies to Equation 18 or to Equation 19 for a single body.

$$\ddot{C} = J^i z^i + J^j z^j \tag{18}$$

$$\ddot{C} = Jz. \tag{19}$$

Multiple constraints can be stacked into the equation system

$$\ddot{C} = \begin{bmatrix} \ddot{C}_1 \\ \ddot{C}_2 \\ \vdots \\ \ddot{C}_m \end{bmatrix}, \tag{20}$$

and can be transferred into a Jacobian matrix $\mathbf{J}$ by partial derivation, if we apply the above scheme.

$$\mathbf{J} = \begin{bmatrix} J_1 \\ J_2 \\ \vdots \\ J_m \end{bmatrix} = \begin{bmatrix} J_1^1 & J_1^2 & \cdots & J_1^n \\ J_2^1 & J_2^2 & \cdots & J_2^n \\ \vdots & \vdots & \ddots & \vdots \\ J_m^1 & J_m^2 & \cdots & J_m^n \end{bmatrix}_{m \times 6N} \tag{21}$$

Each vector J_c^i within the matrix contains six values (sparse matrix). These values are the partial derivatives of the constraint $\ddot{C}_c$ according to the acceleration components of the body i.

4.3 Body Contacts

For an exhaustive simulation of multiple God objects, we have to consider contacts that could occur among objects within the scene. They are also influencing the movement of the God objects. For this purpose, additional contact constraints are taken into account. To avoid the interpenetration of objects, we define that accelerations should always act in the direction of the contact normal. The acceleration at a contact point r is defined as follows.

$$a_r = \underbrace{a_p}_{\text{linear}} + \underbrace{(\alpha \times u_{XYZ})}_{\text{angular}} + \underbrace{\omega \times (\omega \times u_{XYZ})}_{\text{centripetal}} \tag{22}$$

In this equation, u describes the vector in the world frame from the position of the body (center of mass p) to the contact point. Also the linear and angular accelerations, a and α, refer to the world frame. Figure 5 illustrates the acceleration that has to be calculated, so that penetration is avoided in the case of contact. By the quasi-static assumption (see Section 4.2), the angular velocity ω is zero. Therefore, the centripetal acceleration is eliminated. To comply with the normal condition, the acceleration terms are projected onto the contact normal. This sets up the following constraint equation.

$$C(a, \alpha) = an + \alpha(u \times n) \tag{23}$$

The Jacobian matrix can be obtained by partial derivation.

$$\mathrm{J} = \begin{bmatrix} n_x \\ n_y \\ n_z \\ u_y n_z - u_z n_y \\ u_z n_x - u_x n_z \\ u_x n_y - u_y n_x \end{bmatrix}^T_{1 \times 6} \tag{24}$$

A set of c non-penetration conditions for body contacts can be summarized by noting each one in a single row of the Jacobian matrix $\mathbf{J}_{\text{Contacts}}$.

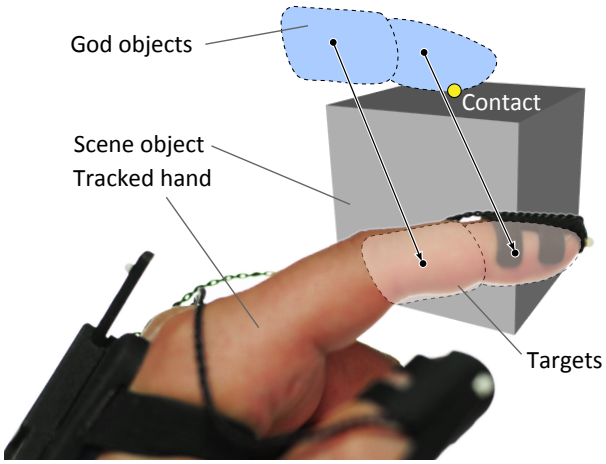

Figure 4: A tracked finger is represented by one God object per phalanx. The finger is penetrating the scene object while the God objects stay on the outside. Through the contact point, a direct manipulation can be computed (i.e. the object might be pushed downwards).

5 The God-Hand Model

We developed the Jacobian matrix for representing a hand model using the components of our extended God-object approach. In analogy to the work of Jacobs et al. [17], our hand model consists of three separate phalanxes per finger. The difference is that each phalanx is represented by a God object rather than by a rigid body. For the palm we use a single God object, resulting in 16 God objects for each hand. Figure 4 shows an example for a single finger. In addition, the individual phalanxes are connected together by joints to simulate the natural movement constraints of the human hand. This limitation could be relaxed if more flexibility is needed (e.g. if different hand sizes of different users are to be used with a single virtual hand model).

To limit the movement of the fingers (F) in relation to the hand's palm (H), we use spherical joints with two degrees of freedom. The five compounds are defined in a Jacobian matrix $\mathbf{J}_{\text{H-F}} \in \mathbb{R}^{20 \times 96}$ containing four constraints for each joint. The matrix has 20 rows given by the number of constraints and 96 columns given by 6D state vectors for each of the 16 hand components.

$$\mathbf{J}_{\text{H-F}} = \begin{bmatrix} \mathbf{J}_{\text{H-F},1} \\ \vdots \\ \mathbf{J}_{\text{H-F},5} \end{bmatrix}_{20 \times 96} \tag{25}$$

In conjunction, the phalanxes are interconnected using revolute joints that allow for just one rotational degree of freedom. As before, the number of rows in the Jacobian matrix depends on the number of constraints. The Jacobian matrix for the constraints of the phalanxes is defined as follows.

$$\mathbf{J}_{\text{F-F}} = \begin{bmatrix} \mathbf{J}_{\text{F-F},1} \\ \vdots \\ \mathbf{J}_{\text{F-F},10} \end{bmatrix}_{50 \times 96} \tag{26}$$

Additionally, we constrain the joint positions to remain within a sensible range according to approximated values reported by Lin et al. [19]. If the joint limits for m constraints are violated within one simulation step, a Jacobian matrix $\mathbf{J}_{\text{Limits}} \in \mathbb{R}^{m \times 96}$ has to be set up. For a complete hand model, the resulting Jacobian matrix $\mathbf{J}_{\text{Hand}} \in \mathbb{R}^{(70+m) \times 96}$ is defined.

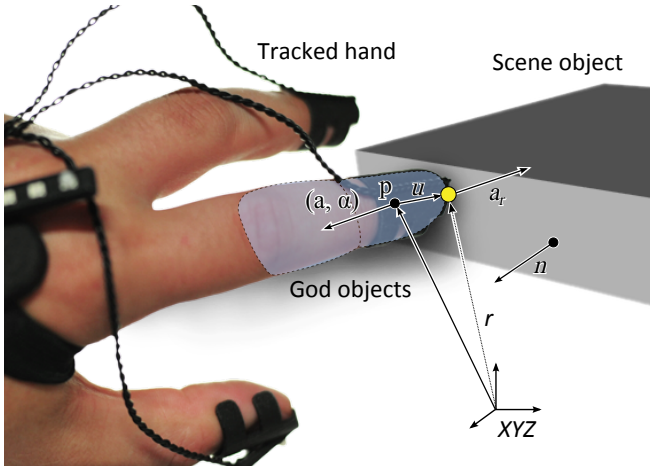

Figure 5: Accelerations acting on a scene object during contact.

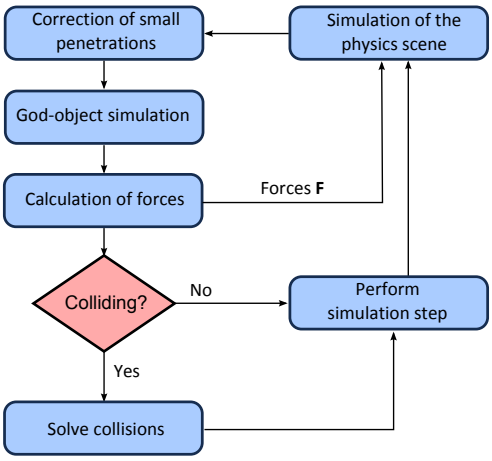

Figure 6: Simulation loop

$$\mathbf{J}_{\text{Hand}} = \begin{bmatrix} \mathbf{J}_{\text{H-F}} \\ \mathbf{J}_{\text{F-F}} \\ \mathbf{J}_{\text{Limits}} \end{bmatrix}_{(70+m)\times 96} \tag{27}$$

The Jacobian matrix for the entire hand model also needs to consider contacts occuring among objects in the scene as introduced in Section 4.3. They are also influencing the movement of the God objects (Figure 5). These additional contact constraints are captured by the Jacobian matrix $\mathbf{J}_{\text{Contacts}}$. The complete Jacobian matrix $\mathbf{J}$ for the hand model just consists of the two stacked Jacobians.

$$\mathbf{J} = \begin{bmatrix} \mathbf{J}_{\text{Contacts}} \\ \mathbf{J}_{\text{Hand}} \end{bmatrix}_{(70+m+c)\times 96}. \tag{28}$$

The motion of the virtual hand can be computed by plugging $\mathbf{J}$ into Equation 9 and solving the resulting system of equations.

Resolving the mutual interdependencies between multiple God objects may lead to large equation systems, which in turn will slow down the God-object solver. The complexity of the equations is highly dependent on the amount of contacts, since each one is represented as an entry in a single Jacobian matrix. To limit the size of the equation systems, we can take advantage of the topological and geometric properties of the scene. Connected God objects such as the fingers of a hand should be clustered together. Completely independent God objects, such as the two hands of a user, should be organized as separate clusters. The corresponding equation systems are independently solved based on the assumption that separate clusters do not share any contact points and do not directly interact with each other. Nevertheless, the interaction of both hands with a single object in the scene is still possible as discussed in Section 7.

6 Implementation

We describe the implementation of the presented concepts for the interaction of two God hands with a physically simulated virtual world.

6.1 Handling of Complex Scenes

In the automotive context, virtual assembly simulations often require the use of highly detailed models, which quickly make the continuous collision detection a bottleneck. As a result, only low frame rates are achieved. Hence, we had to consider a trade-off between simulation quality and interactive simulation rates. The calculations using the God-object method are very accurate and provide a penetration-free state for any number of objects. However, a penetration-free simulation is particularly important for the objects with which the user directly interacts. Typically scene objects move rather slowly, whereby they generally only undergo small interpenetrations. This fact is exploited in order to save simulation resources. We decided to simulate only the hands using our extended God-object method. The remaining scene objects are simulated by a less precise but much more efficient dynamics solver. As a consequence, a coupling of the two solvers is required. The decisive factor is the required exchange of forces between the two simulations such that a consistent state is reached. This approach extends the scope of our approach to larger models that can be simulated in real time.

The library TSNNLS [11] is used to calculate the minimization system for the God objects. We use the CLAPACK [32] library for the factorization and the inversion of matrices. The latter is also used by the extended God-object implementation for solving linear systems of equations and performing matrix multiplication. For the CCD we are using the FAST library [34].

6.2 Simulation Loop

Figure 6 shows the required steps for coupling our extended God-object approach and the dynamic simulator for scene objects. Hereafter scene objects are referred to as physics objects to clarify the distinction of God objects.

We use the physics engine *Bullet* [31] for handling our physics objects. As a first step, a complete calculation through the physics pipeline of *Bullet* is performed, including collision detection, solver calculations as well as pose and force updates. This will include forces from the God objects calculated in the previous time step. All objects are in a valid state after the simulation by the physics engine *Bullet*.

The movement of the physics objects can cause slight penetrations with the God objects. Therefore, in the next step a collision detection has to be performed to produce a penetration-free state. In analogy to the descriptions of Millington [20], a non-linear projection solves slight penetrations between a pair of objects considering plausible object movements. This step is necessary so that the computation of contact points for the extended God-object approach provides sensible results. It should be noted that such a correction can produce new penetrations among other objects. Therefore, the correction of slight penetrations needs to be performed iteratively until a penetration-free state is reached for all pairs of physics objects or a maximum number of iterations is reached. The deepest intersections are resolved first in order to reduce the likelihood of new penetrations.

What follows is the simulation of the God objects, and in our case we consider two God hands. This computation is done independently for each hand, since collisions between the hands are undesirable in most scenarios. Under this condition, following the recommendation of Section 5, the equation systems to determine the accelerations can be kept smaller and executed sequentially. The necessary steps for calculating the accelerations for a set of God objects was described in detail in Section 4. The resulting forces are passed on to the physics engine *Bullet*.

The integration step is performed after having calculated the generalized constrained acceleration vector z_c for all God objects. In this case, starting from the last time step t_0, a new target position for a God hand at the current time $t_0 + \Delta t$ is calculated with Δt being the length of the current time step. The integration is performed through a symplectic Euler step in generalized coordinates. The integration step in the last part of the simulation loop calculates the target transformations for the God objects that are assumed based on the body accelerations. All constraints that were in existence at the beginning of the time step are included in the motion of the bodies by determining their constrained acceleration.

The calculation of the target transformations for God objects can in turn lead to intersections with the physics objects. Therefore, which God objects would be in contact with physics objects after the application of the target transformation needs to be evaluated once again. We perform a CCD for all God objects that are in a contact state with physics objects. The CCD computes the earliest TOC along the trajectory of God objects. God objects have to adapt their target transformation if the CCD returns a collision to maintain a penetration-free state. This collision handling is performed by the algorithm that is described in the following section.

6.3 Collision Handling

Collision handling is critical for the interaction of the two solvers. An overview is provided in Algorithm 1. The God objects $A^1, A^2, A^3, ..., A^{16}$ and the physics objects B^j are considered as a starting point for continuous collision detection.

At the beginning, only the movements of the God objects during the current time step are considered since the movements of God objects and physics objects are performed successively. Thus the components A^i are moving from t_0 until $t_0 + \Delta t$. We normalize the time interval Δt to the interval $t = [0,1]$ for simplicity. After the integration step the movement of an object A^i is known and it is transformed from $T_{A^i,t=0}$ to the transformation $T_{A^i,t=1}$. For an object B^j, no movement is allowed during the God-object simulation resulting in $T_{B^j,t=0} = T_{B^j,t=1}$.

The transformations and the geometry of all pairs of objects (A^i, B^j) are passed individually to the CCD. The CCD calculates the TOC at which objects collide. The computation is based on the assumption that the objects do not leave their trajectories. The positions of God objects have to be corrected if a collision was found. During the symplectic Euler step, the time step Δt is shortened to the minimal TOC of all God objects, which just results in a multiplication of Δt with the minimal TOC due to the normalization (Equations 29 and 30).

$$y^i(t_c) = z^i(t_0)\,\Delta t\,\mathrm{TOC}_{\min} \tag{29}$$

$$x^i(t_c) = x^i(t_0) + y^i(t_c)\,\Delta t\,\mathrm{TOC}_{\min} \tag{30}$$

Finally, the contact points must be generated for the time step t_c and stored for the God objects. Contact points are removed that are no longer active. The remaining contact points are treated as constraints during the next time step. After updating the pose information, the transformations of the virtual hands and the physics objects are stored and can be passed on to the rendering system for display. Afterwards, the simulation loop repeats for the next time step.

Algorithm 1 collision handling

```
1:  TOC_min ⇐ 1
2:  for all God objects A do
3:      T_start ⇐ start transformation of A
4:      T_target ⇐ target transformation of A
5:      for all physics objects B do
6:          T_B ⇐ transformation of B
7:          Result R = CCD(A, T_start, T_target, B, T_B) // collision detection
8:          TOC_min ⇐ min(TOC_min, R.TOC)
9:      end for
10: end for
11: for all God objects A do
12:     move A until TOC_min
13:     update contacts for time step TOC_min
14: end for
```

7 Results

To explore the principal behavior of our implementation, we performed a number of benchmarks on a single computer equipped with an Intel Core i7-940. The test scene was displayed in a three-sided CAVE as shown in Figure 7. Rendering for the CAVE was done by a cluster, which only receives the position updates from the physics simulator. The physics engine *Bullet* was used to simulate the interaction among the objects in the scene.

We observed that our algorithm always provided a non-penetrating visual representation of the user's hand independently of how far a user stuck the hands into an object. Contact constraints are therefore treated correctly by the simulator. The combination of both solvers, for God objects and the physics scene, worked without major difficulties. Even rapid movements of the user were handled correctly by our hybrid approach. While the penetrations were correctly avoided, which was our goal in the first place, there were some situations where the simulation did not fully respect all of the constraints created by the joints of the God hands. This led to slightly separated finger phalanxes. A higher prioritization of joint constraints and their limits within the optimization problem could ameliorate this problem.

Figure 8 shows some benchmark results for a short interaction sequence with one God hand within our basic test scenario (Figure 7a). The simple test scene contained only five physics objects. The total number of triangles in the scene was 100,000 and the largest object consisted of about 30,000 triangles. It can be clearly seen that the computation time directly depends on the number of contact points in each simulation step. The CCD algorithm generates many contact points, which is necessary to achieve a stable simulation. Fortunately we observed only a linear relationship between the number of constraints and processing time, which was also reported by Baraff [3]. During the entire interaction sequence

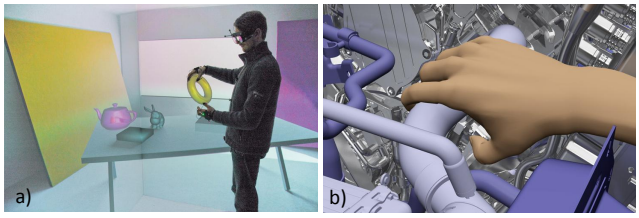

Figure 7: a) A simple test environment for finger-based interaction in a CAVE. b) Penetration-free collision between a God hand and a detailed engine compartment of a car consisting of one million triangles.

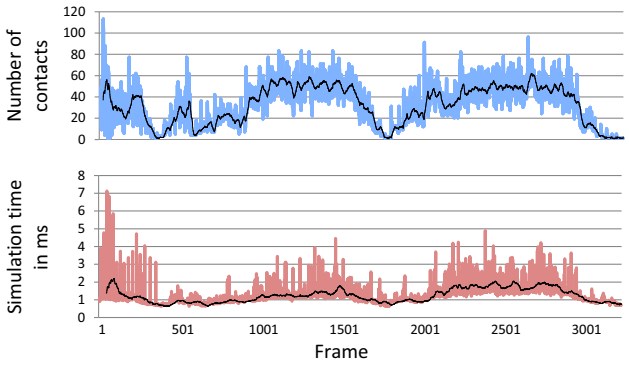

Figure 8: Benchmark simple scene – contacts during an interaction sequence and corresponding simulation time over a sequence of frames.

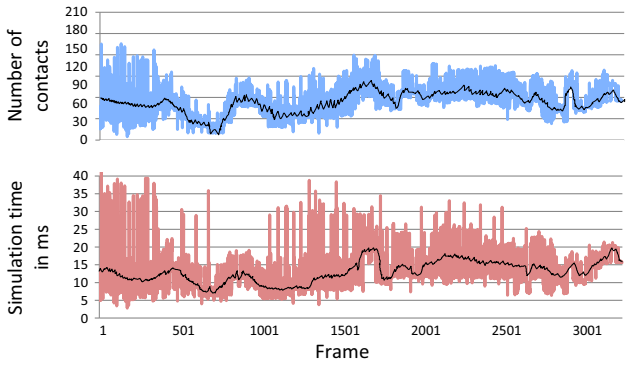

Figure 10: Benchmark motor compartment – timings vary more strongly than for the simple scene.

we achieved an interactive simulation rate of at least 100Hz. On average, a value of more than 600Hz was achieved. The calculation peaks correlate directly with points in time when a large number of contacts was found. A more detailed analysis of the benchmark results provides information about the optimization potential. Figure 9 shows the breakdown of the computation time for the contributing parts of a simulation step. It is clearly evident that the continuous collision detection contributes a large fraction of about 40% of the frame time. There is also further potential for optimization by thinning out overly dense contact points.

We also performed benchmarks involving two-handed interactions. For each God hand, an independent cluster of God objects was created and collisions between both hands were not considered. To avoid slight penetrations caused by separated calculations of the accelerations, we are using the collision handling method as described in Section 6.3. In this step, all penetrations are solved algorithmically by taking all God objects into account. During an interaction sequence with two hands, we achieved a speed-up for the calculation of accelerations by a factor of approximately 1.7, whereas the penetration solving process was as fast as with one calculation cluster for both hands.

We also investigated more complex models as shown in Figure 7b. The engine compartment consists of about 1,000,000 triangles. The largest single object was made up of about 100,000 triangles. In total, the scene consists of approximately 20 accessible parts that are simulated. The simulation rate typically varied between 50Hz and 70Hz depending on which part of the engine was touched. However, it could also drop below 20Hz if a large number of contacts was generated in the proximity of a highly detailed part. Figure 10 shows the simulation results of the compartment scene in a one-handed interaction sequence. In comparison to the simple test scene, we observe a much stronger variation of the overall simulation time. During peaks, the CCD utilizes up to 80% of the processing time even though the number of generated contact points is not much larger in this scene as it mostly depends on the God-hand model.

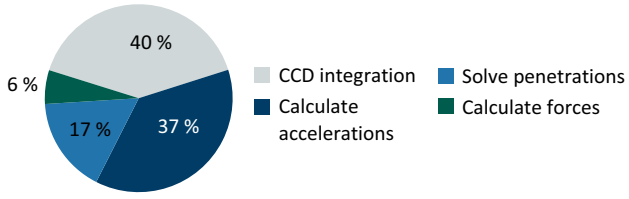

Figure 9: Analysis of the contributions of different simulation components to the duration of a simulation frame within the simple test scene.

We explored the limitations of our approach by performing some tests with current CAD models in which individual parts consisted of up to 400,000 triangles. In this case the CCD utilizes about 70-80% of the simulation time. The overall performance was still interactive at about 10-15Hz. We expect that replacing the general purpose CCD implementation of the FAST library [34] by an adapted and parallelized implementation will considerably extend the applicability of our system.

8 Conclusions and Future Work

We extended the 6-DOF God-object method to simultaneously consider multiple dependent and independent God objects. First we derived an equation system for simulating multiple God objects using Gauss principle of least constraint and showed that it is a non-negative least squares problem, which can be efficiently solved. We also showed how to consider several types of constraints during the optimization. Based on our general formulation for multiple God objects, we derived a description of a God-hand model consisting of a palm and separate connected phalanxes for each finger. For dealing with complex virtual environments, we coupled our precise God-hand simulation to a physics engine, which simulates the motion of regular objects in the scene. The experience with our implementation showed that the God-hand simulation is very stable, effectively avoids deep virtual hand-object penetrations, generates plausible finger positions and runs at interactive frame rates for models consisting of more than one million triangles. In addition, motion artifacts such as artificial friction, pop-through effects or fingers getting stuck in an object were avoided.

The central goal of this work was to avoid deep penetrations of a virtual hand model and objects in the scene. This is an important requirement for adding a stable friction simulation into the model as a next step for guaranteeing stable grasps. The approach by Duriez [13] allowed rather precise simulation of friction behavior, but it led to a non-linear complementarity problem (NCP), which is computationally expensive if the number of contact points gets large. However, it is promising that we only need to deal with the contact points of the God hands. An alternative is the use of a soft-body representation of the user's fingers in combination with our generalized God-object approach, which also has the potential for a precise and direct manipulation of virtual objects by the user's hands.

ACKNOWLEDGEMENTS

We thank the team of the Volkswagen VRlab for their help with the hardware setup and the valuable discussions, Raimund Dachselt from the User Interface & Software Engineering Group at Otto-von-Guericke-Universität Magdeburg for his support and the reviewers for their detailed and constructive comments.

REFERENCES

[1] D. Baraff. Analytical methods for dynamic simulation of non-penetrating rigid bodies. In *Proceedings of the 16th annual conference on Computer graphics and interactive techniques*, SIGGRAPH '89, pages 223–232, New York, NY, USA, 1989. ACM.

[2] D. Baraff. Coping with friction for non-penetrating rigid body simulation. In *Proceedings of the 18th annual conference on Computer graphics and interactive techniques*, SIGGRAPH '91, pages 31–41. ACM, 1991.

[3] D. Baraff. Fast contact force computation for nonpenetrating rigid bodies. *Proceedings of the 21st annual conference on Computer graphics and interactive techniques*, pages 23–34, 1994.

[4] J. Bender, D. Finkenzeller, and A. Schmitt. An impulse-based dynamic simulation system for VR applications. In *Proceedings of Virtual Concept 2005*, Biarritz, France, 2005. Springer.

[5] J. Bender and A. Schmitt. Constraint-based collision and contact handling using impulses. In *Proceedings of the 19th international conference on computer animation and social agents*, pages 3–11, Geneva (Switzerland), July 2006.

[6] M. Bergamasco, P. Degl'Innocenti, and D. Bucciarelli. A realistic approach for grasping and moving virtual objects. *Proceedings of IEEE/RSJ International Conference on Intelligent Robots and Systems IROS94*, 1:717–724, 1994.

[7] Å. Björck. *Numerical Methods for Least Squares Problems*. Society for Industrial Mathematics, Philadelphia, 1996.

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

[9] R. Bro and S. De Jong. A fast non-negativity-constrained least squares algorithm. *Journal of Chemometrics*, 11(5):393–401, 1997.

[10] E. Burns, S. Razzaque, A. T. Panter, M. C. Whitton, M. R. McCallus, and F. P. Brooks, Jr. The hand is more easily fooled than the eye: users are more sensitive to visual interpenetration than to visual-proprioceptive discrepancy. *Presence: Teleoperators & Virtual Environments*, 15:1–15, February 2006.

[11] J. Cantarella and M. Piatek. Tsnnls: A solver for large sparse least squares problems with non-negative variables. *CoRR*, cs.MS/0408029, 2004. `http://www.jasoncantarella.com/`.

[12] J. Choi, J. Dongarra, L. Ostrouchov, A. Petitet, D. Walker, and R. Whaley. Design and implementation of the ScaLAPACK LU, QR, and Cholesky factorization routines. *Scientific Programming*, 5(3):173–184, 1996.

[13] C. Duriez, H. Courtecuisse, J. Alcalde, and P. Bensoussan. Contact skinning. In *Proceedings of the 2008 Eurographics*, volume 27, 2008.

[14] K. Erleben. Stable, robust, and versatile multibody dynamics animation. *Unpublished Ph. D. Thesis, University of Copenhagen, Copenhagen*, 2004.

[15] C. Gauß. Über ein neues allgemeines Grundgesetz der Mechanik. *Journal für die reine und angewandte Mathematik*, 1829(4):232–235, 1829.

[16] D. Holz, S. Ullrich, M. Wolter, and T. Kuhlen. Multi-contact grasp interaction for virtual environments. *Journal of Virtual Reality and Broadcasting*, 5(7), July 2008. ISSN 1860-2037.

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

[18] S. Kockara, T. Halic, C. Bayrak, K. Iqbal, and R. Rowe. Contact Detection Algorithms. *Journal of Computers*, 4(10):1053, 2009.

[19] J. Lin, Y. Wu, and T. Huang. Modeling the constraints of human hand motion. *Proceedings of the Workshop on Human Motion (HUMO'00)*, pages 121–126, 2002.

[20] I. Millington. *Game physics engine development*. Morgan Kaufmann, 2007.

[21] B. V. Mirtich. *Impulse-based dynamic simulation of rigid body systems*. PhD thesis, University of California, Berkeley, 1996.

[22] 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. IEEE, 2010.

[23] M. Moehring and B. Froehlich. Effective Manipulation of Virtual Objects Within Arm's Reach. In *Virtual Reality Conference (VR), 2011 IEEE*, pages 131–138. IEEE, March 2011.

[24] M. Moore and J. Wilhelms. Collision detection and response for computer animation. *Proceedings of the 15th annual conference on Computer graphics and interactive techniques*, pages 289–298, 1988.

[25] M. Mueller. Hierarchical position based dynamics. In *Proceedings of Virtual Reality Interactions and Physical Simulations (VRIPHYS)*, pages 1–10, 2008.

[26] M. Mueller, B. Heidelberger, M. Hennix, and J. Ratcliff. Position based dynamics. *Journal of Visual Communication and Image Representation*, 18(2):109–118, 2007.

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

[28] M. Ortega, S. Redon, and S. Coquillart. A six degree-of-freedom god-object method for haptic display of rigid bodies with surface properties. *IEEE Transactions on Visualization and Computer Graphics*, pages 458–469, 2007.

[29] S. Redon, A. Kheddar, and S. Coquillart. Gauss' least constraints principle and rigid body simulations. *Robotics and Automation, 2002. Proceedings. ICRA'02. IEEE International Conference on*, 1:517–522, 2002.

[30] M. Weber, G. Heumer, H. B. Amor, and B. Jung. An animation system for imitation of object grasping in virtual reality. In *ICAT*, pages 65–76, 2006.

[31] Website. *Bullet Physics Library*, Nov. 2011. `http://bulletphysics.org`.

[32] Website. *CLAPACK*, Nov. 2011. `http://www.netlib.org/clapack/`.

[33] 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.

[34] X. Zhang, M. Lee, and Y. Kim. Interactive continuous collision detection for non-convex polyhedra. *The Visual Computer*, 22(9):749–760, 2006.

[35] C. Zilles and J. Salisbury. A constraint-based god-object method for haptic display. *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems 95*, 3:146–151, 1995.