Haptic Interaction with a Glove Interface in a Physics Based Virtual Environment

Michael F. Zaeh*, Hans Egermeier*, Bernd Petzold*, and Harald Schmid#
*Institute for Machine Tools and Industrial Management (*iwb*), Technische Universtiät München, 85748 Garching, Germany
{michael.zaeh, hans.egermeier, bernd.petzold}@iwb.tum.de
#Technische Universtiät München, 85748 Garching, Germany
schmid@manageandmore.de

Abstract

In order to enhance productivity and to speed up the ramp-up of manual production processes, manual assembly simulations are an important step during early design and planning stages. One of the most essential requirements is the realistic dexterous object manipulation. An intuitive approach is the use of Virtual Reality (VR) techniques in combination with force feedback devices, in order to enhance the visual representation and the user interaction with the virtual prototypes. The presented concept is a new approach combining the GJK algorithm for collision detection with linear complementarity techniques to model and simulate multi-rigid-body dynamics with contact and friction between the user's hand and the manipulated objects. The concept is implemented to support a stable fine object manipulation in the simulation environment Virtual engineering environment Ve², which is currently developed at the iwb.

Key words: Haptic, Glove Interface, Dexterous Manipulation

1. Introduction

In manual assembly simulations, object manipulation is the most essential man-machine interaction between the virtual environment and users. A promising approach is the use of haptic interfaces for dexterous object manipulation. With a virtual scene haptic interfaces add a sense of touch to interactions. The sensations of touch are generated by detecting collisions and deducing forces from the collision situation between user and objects in the virtual environment (VE) and reflecting those forces back to the user.
For this reason a VR-system is developed at the *iwb* with the focus on man-machine interaction to allow realistic and meaningful assembly simulation. The Virtual engineering environment Ve² is built up by different modules. Asides from a number of modules that enable Ve² to be configured, to integrate different device drivers, and to connect to remote processes, the three core modules are:

- **Visualisation module**: this module is responsible for the graphical representation of the virtual environment. It supports different displays from monitor screens up to full immersive multi-projection walls.
- **Haptic rendering module**: the haptic rendering is needed to detect contacts and intersections in the scene and to calculate the penetration depth vectors and contact points.
- **Rigid-body simulation module:** the simulation module computes the object movements based on the detected contacts, the kinematic constraints and the exerted torques and forces.

This paper focuses on the haptic rendering and the rigid-body simulation module, because these two modules are mainly responsible for an appropriate haptic interaction. A lot of research was done in the field of collision detection reducing the computational effort and enhancing the used algorithms for contact, local minimum distance and penetration depth computation based on polygonal, voxel, and point shell models for either single or multiple haptic interface points. But for manual assembly simulations not only the collision detection itself is important. Realistic physical object behaviour and, realistic object manipulation with the virtual user's hands is essential for the simulation as well. Although for rigid-body simulation techniques as much research was done as for collision detection approaches, combinations of an efficient collision detection, physically based object behaviour, and dexterous object manipulation with a force reflecting glove interface are only rarely reported.
This paper describes a new possibility to integrate a collision detection based on a stable implementation of the GJK algorithm, a fast and stable physics simulation using linear complementarity techniques to simulate multi-rigid-body dynamics with contact and friction, and a haptic hand interface.

2. Main Results

The presented concept shows the combination of open source software packages for collision detection and physical simulation to support dexterous object manipulation with a force reflecting glove interface. Using geometric primitives for the approximation of the virtual hand allows a fast collision detection and contact computation of the hand model. The developed hand simulation interface provides a stable force and torque computation algorithm for hand object interactions, in order to support dexterous manipulation of objects. The concept is realised in the VR-system Ve² which is used as testbed to evaluate the concept and the algorithms.

3. Previous Work

Implementations for dexterous object manipulation strongly depend on the ability to show realistic motion control of the manipulated object. Collision detection plays a key role. It is necessary to prevent objects from interpenetrating and to provide the basic information that can be used to compute geometric constraints or contact forces involved in the manipulation process. A lot of research has already been conducted in the field of collision detection and therefore many different approaches and implementations are known. Besides using voxel [1] or point shell [2] based algorithms, the use of polygon based approaches is the most common. An evaluation for some available implementations of polygon based algorithms is given in [3]. One of the tested implementations is SOLID. It employs the Gilbert-Johnson-Keerthi (GJK) algorithm, which is reported first in [4]. SOLID also makes use of modifications explained in [5] and [6] to provide results faster and with less instability. In most of the tests described by Reggiani SOLID is rated well among the other tested libraries. In addition, Luciano [7] recently reported that SOLID is suited very well as collision detection for haptic applications.

The second key role to show realistic object behaviour is the real-time computation of rigid-body dynamics. The well known basics of rigid-body dynamics are presented for instance in [8] and [9]. Although many implementations of rigid-body dynamics simulation software use collision detection in order to compute contact information, it is only rarely reported that this information is used to support haptic interaction devices. One reason for that are stability issues, which are for instance discussed in [10]. Ikei [11] presents a method, where the behaviour simulation method of manipulated objects is based on statics, in order to reduce the complexity of real-time computation. In this case, it is shown that statics calculation can provide sufficiently realistic behaviour of the manipulated objects. It is assumed that the user grasps the manipulated object at one point. The haptic interaction device and the object are then connected by a virtual spring. Grasping and ungrasping is realised by a separate hardware switch on the input device. Recently an implementation of an integrated haptics and dynamics simulation was published by Hasegawa [12], [13]. The software called SPRINGHEAD proposes a method for real-time rigid-body simulations for haptic interactions based on a penalty method regarding the contact volume of intersecting bodies. The method is tested with a haptic interface controlling, a so called haptic pointer.

The third important aspect of dexterous manipulation is the integration of collision detection, physical simulation and hand models, which allow realistic user interaction. Kijima [14] reports an approach of object manipulation, where the interacting part of the virtual hand is reduced to three fingertips. Two example manipulation calculations with fingertips in VE without force feedback are developed. The two methods presented are called ”Impetus Method” and ”Representative Spherical Plane Method”. The first method is based on the collision between only one fingertip and the object surface. It belongs to the category of semi-dynamics. The second method is based on the restricted interaction among three fingertips and a sphere that represents the object. It belongs to the field of kinematics. Experimental results show the superiority of object manipulation to conventional gesture based methods.
A more sophisticated hand model, which represents the whole user’s hand is described by Boulic [15]. Spheres placed on finger joints act as sensors to detect collisions, which have shown to be suitable for dexterous object manipulation. The aim of the proposed method is to overcome the lack of force feedback of general purpose digital gloves. The aim is to derive a visual restitution consistent with the user manipulative intentions while respecting the integrity of solid interactions with friction. No rigid-body dynamics are involved in this approach. Huang et al. [16] use a similar method, which uses spheres as sensors placed on finger joints to detect collisions.
In [17] Furusawa describes a peg in a hole assembly scenario for two collaborating hands. The peg is controlled by one hand using a haptic display called ”HapticMaster”. The part, where the peg needs to be inserted, is controlled by a data glove interface without force feedback. The user’s hand is represented as a set of bounding boxes. The grasping algorithm is only a function of the number of intersections of bounding boxes of the fingers and the manipulated objects. This model neither supports fine manipulation with the fingers nor rigid-body dynamics.
Hirota [18] describes a constraint based approach to calculate contact forces. The proposed method simulates physical manipulation of objects with force feedback. The interaction force is computed based on the constraint that is caused by the object on the user’s finger. It expands the common concept of the god-object method, so that physical constraints during the spatial motion of haptic interface points can be simulated. This concept is implemented to support two fingers connected to force feedback devices. This concept was further developed by Hirota [2] to a point shell concept for the whole hand, which supports user interaction with a glove interface. The improved concept allows fine

manipulation of physical simulated objects, currently without force feedback.

4. Algorithms and Concepts

The virtual world scene simulated by Ve² consists of three different spaces. A visual space for the visual representation of the objects, a collision space for collision detection and contact computation, and a physics space for the computation of the rigid-body dynamics. The representation of an object in a space is called body. The coherence between the three spaces is maintained through synchronisation.

A. Hand Model

The representation of the user's hand consists of a kinematics skeleton, which is updated by the device driver of the haptic glove interface. It consists of fifteen finger joints and a palm, with altogether twenty-two degrees of freedom. This skeleton is coupled with the

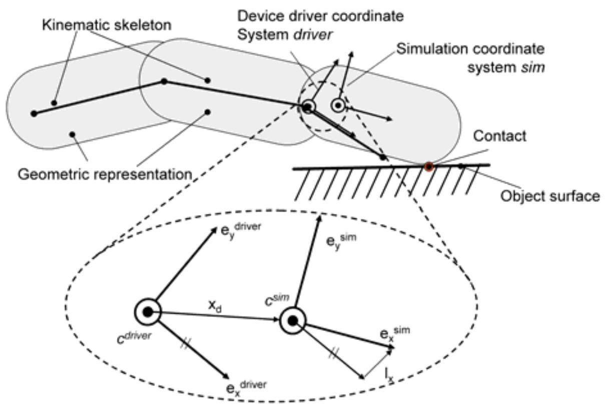

Fig. 1 Schematic view of the implemented constraint mechanism of the hand model

geometric and physical hand representation over spring-damper elements, generating the corresponding limb constraint forces $\mathbf{f}_{lc}$ and torques $\mathbf{t}_{lc}$ to synchronise the skeleton with each geometric and physical representation of all finger limbs and the palm. See Eqn. 1 for the constraint force computation based on the position deviation vector $\mathbf{x}_d$ and the deviation speed $\mathbf{v}_d$, and Eqn. 2 for the constraint torque computation based on the deviation vector $\mathbf{t}_d$ and the rate of change of the deviation vector $\Delta\mathbf{t}_d$. $\mathbf{t}_d$ is used as measurement of the difference of the orientation of the simulated hand part coordinate system c^{sim} relatively to device driver coordinate system c^{driver}, see Fig. 1. k_f and k_t are the stiffness factors, d_f and d_t are the damping factors. Although penalty methods to oppose constraints, which add extra energy to the system and tend to stiff differential equations and lower accuracy, this approach is considered to be sufficient to enable a stable and realistic hand interaction.

$$\mathbf{f}_{lc} = k_f \cdot \boldsymbol{x}_d - d_f \cdot \boldsymbol{v}_d \tag{1}$$

$$\mathbf{t}_{lc} = k_t \cdot \mathbf{t}_d - d_t \cdot \Delta\mathbf{t}_d \tag{2}$$

with $$\mathbf{t}_d = \sum_{i=x,y,z} \mathbf{l}_i \times \mathbf{e}_i^{sim} \tag{3}$$

with $$\mathbf{l}_i = \mathbf{e}_i^{sim} - \mathbf{e}_i^{driver} \qquad i = x, y, z \tag{4}$$

The visual body of the hand is represented as a polygonal model where every single limb including the palm is a separate geometry.The collision body of the virtual hand is modelled as Minkowski sum built up out of geometric primitives. Each finger limb is represented as the Minkowski sum of a sphere and a line segment. The resulting body is a capped cylinder, where the length and thickness of the cylinder is determined by the

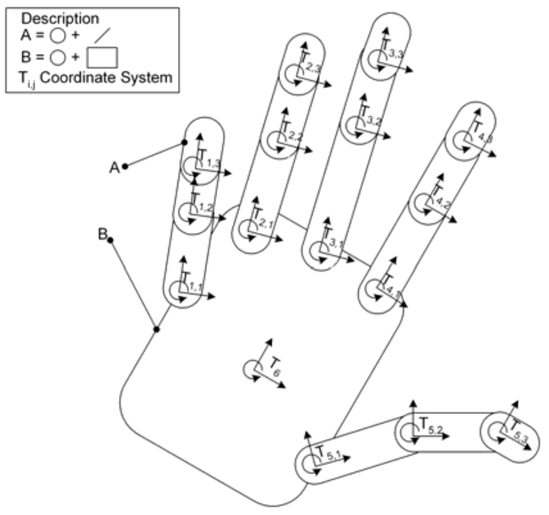

Fig. 2 Approximation of the users hand for the collision detection by the Minkowski sum out of primitives

length of the line and the radius of the sphere. The palm is represented as the Minkowski sum of a sphere and a box. The resulting body is a box with round edges, depending on the radius of the sphere, see Fig. 2. The Minkowski sum *A+B* of two sets *A* and *B* is defined as:

$$A, B \subset R^3, \quad A + B := \{a + b; a \in A, b \in B\}. \tag{5}$$

Physically the hand limbs are modelled as mass located at the origin of the limb coordinate system and an inertia matrix, which describes how the mass is distributed around the centre of mass. The geometric dimensions of the physical representation are determined by the dimensions of the corresponding collision body. The position and orientation of the physical limbs also specify the geometric representations for visualisation and collision detection.

B. Object Representation

Ordinary objects, other than the hand, have exactly the same polygonal representation for the visual body and for the collision geometry. No simplifying assumptions are made. In the collision engine, objects are represented as compound of convex objects. Their physical representation is characterised by the body's mass and the inertia tensor centered at the body's origin.

C. Grasping Algorithm and Force Computation

In order to manipulate touched or grasped objects in a realistic way, contact forces between the virtual hand and the objects in the scene need to be calculated. The following steps are computed, see Fig. 3.

The first step of the algorithm is the collision detection. Based on the GJK algorithm described in [6] the collision detection and penetration depth computation problems are expressed in terms of the configuration space obstacle (CSO). For this purpose a negation operation of the Minkowski sum need to be indroduced

$$-B = \{-b : b \in B\}, \tag{6}$$

so that the CSO can be written as *A + (-B)* which will be abbreviated to *A - B*. Following from that the collision detection problem is expressed as

$$A \cap B \neq \emptyset \Leftrightarrow \mathbf{0} \in A - B\,, \tag{7}$$

where **0** denotes the origin of the configuration space. Therefore, the magnitude of the penetration depth *p(A,B)* can be expressed as

$$p(A,B) = inf\{\|\mathbf{x}\| : \mathbf{x} \notin A - B\}, \tag{8}$$

where *p(A,B)* denotes the shortest distance from **0** to the boundary of *A-B*. See Fig. 4 for the relation between intersecting objects and their CSO and the resulting penetration depth.

The next step of the grasping algorithm is the contact definition. If one or more intersections between the hand and objects are detected, a contact is defined at each position where the hand intersects the objects. A contact is specified by the direction of the penetration depth vector *p(A,B)*, the contact position, the friction coefficient μ, and the ODE specific parameters *ERP* (error reduction parameter) and *CFM* (constraint force mixing). *ERP* specifies what proportion of simulation errors regarding joints and contacts will be fixed during the next simulation step. The *CFM* parameter specifies the amount, with which the original constraint equation of the rigid-body system can be violated. With these two parameters the surface characteristics are expressed with regard to surface stiffness and surface damping. Furthermore, the contact situation between two or more objects is computed based on the Coulomb friction model

$$|\mathbf{F}_T| \leq \mu|\mathbf{F}_N|, \tag{9}$$

where $\mathbf{F}_T$ and $\mathbf{F}_N$ are the tangetial and normal force vectors and μ is the friction coefficient. Due to efficiency reasons, this friction model is approximated to

$$\mathbf{F}_m = \mu|\mathbf{F}_N|, \tag{10}$$

where $\mathbf{F}_m$ denotes the maximum force limit.

During the next step the equations of motion of the rigid-body system are solved. The constraint equations of the system are shown in Eqn. 11, which are solved with a Danzig LCP solver, see also [20] and [21] for a detailed description. Eqn. 12 is the ODE specific

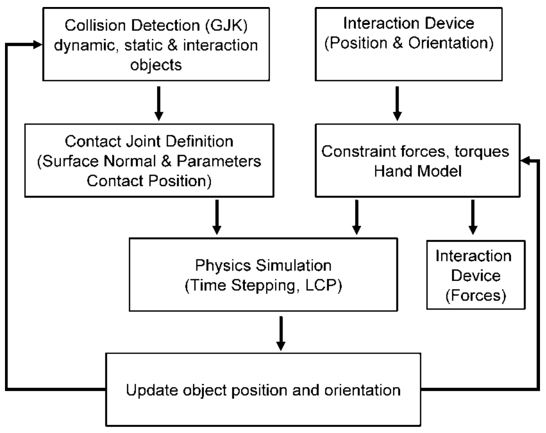
Collision Detection (GJK) dynamic, static & interaction objects
Interaction Device (Position & Orientation)
Contact Joint Definition (Surface Normal & Parameters Contact Position)
Constraint forces, torques Hand Model
Physics Simulation (Time Stepping, LCP)
Interaction Device (Forces)
Update object position and orientation

Fig. 3 Simulation sequence of grasping algorithm and

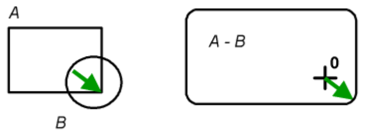

Fig. 4 A pair of intersection objects and their corresponding CSO. The arrow represents the penetration depth

modification [19] of Eqn. 11, where **J** is the constraint Jacobian matrix, **M** the system's mass matrix, λ the Lagrange multipliers the equation needs to be solved for, *h* the timestep stepsize of the simulation, and **c** the vector of the forces being applied to the bodies.

$$\mathbf{JM}^{-1}\mathbf{J}^T\lambda = c \tag{11}$$

$$\left(\mathbf{JM}^{-1}\mathbf{J}^T + \frac{CFM}{h}\right)\lambda = \frac{c}{h} \tag{12}$$

Solving Eqn. 12 leads to the constraint forces $\mathbf{f}_c$ which preserve the system constraints.

$$\mathbf{f}_c = \mathbf{J}^T\lambda \tag{13}$$

Knowing all the constraint forces and external forces and torques exerted to the bodies, which are summed up in **F** and **T**, the change in linear momentum Δ**P** and angular momentum Δ**L** can be calculated.

$$\mathbf{F} = \Delta t = \Delta\mathbf{P} \qquad \mathbf{T}\Delta t = \Delta\mathbf{L} \tag{14}$$

Following from that the new linear and angular velocity $\mathbf{v}_t$ and ω_t of the current timestep *t* of each body can be computed, where $\mathbf{I}^{-1}$ is the inverse inertia tensor.

$$\mathbf{v}_t = \mathbf{v}_{t-1} + \frac{\Delta \mathbf{P}}{m} \qquad \omega_t = \omega_{t-1} + \mathbf{I}^{-1}\Delta \mathbf{L} \quad (15)$$

As last step of the rigid-body simulation the position $\mathbf{x}_t$ and orientation $\mathbf{q}_t$ of the bodies is updated.

$$\mathbf{x}_t = \mathbf{x}_{t-1} + \Delta t \mathbf{v}_t \quad (16)$$
$$\dot{\mathbf{q}}_t = \tfrac{1}{2}\omega \mathbf{q}_{t-1} \quad (17)$$
$$\mathbf{q}_t = \mathbf{q}_{t-1} + \Delta t \dot{\mathbf{q}}_t \quad (18)$$

After solving the rigid-body system, the position and orientation of all objects, which are defined in the visual space and the collision space, are updated as well. From the resulting difference of the new position and orientation, calculated by the rigid-body simulation and the device driver update, the constraint forces of the hand model are computed according Eqn. 1 and Eqn. 2. The calculated forces are then sent to the haptic hand interface. Also, the hand constraint forces and torques are saved for the next simulation step, so that the differences can be corrected by the rigid-body simulation.

5. Implementation and Results

The currently used hardware setup of the described system is shown in Fig. 5. The used workstation is an SGI Onyx 2 running four 250 MHz IP27 processors of the type MIPS R10000 and with a two GB main memory. The graphics board is of the type InfiniteReality2E. The used operating system is IRIX

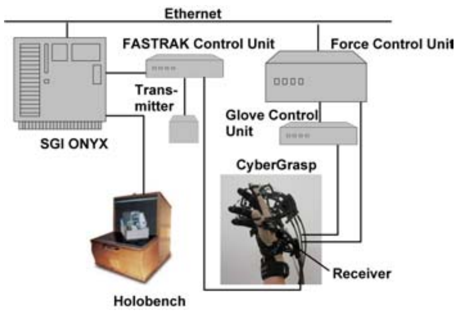

Fig. 5 Hardware setup of Ve²

6.5. The workstation is directly connected to a FASTRAK control unit with two receivers: one used for head tracking and the other used for the position tracking of the user's hand. As display, a two sided projection system is used. The CyberGrasp Force Control Unit (FCU) is connected to the workstation via Ethernet. The basic runtime structure of the used VR-system Ve² consists of three different loops controlling the application. The display loop runs with 20 Hz and is solely responsible for graphical rendering. The collision and simulation loop runs depending on the tested models with 230 up to 250 Hz and is responsible for collision detection and executes the simulation engine in every frame. The kernel loop is synchronised with the collision and simulation loop and runs the same frame rate. This

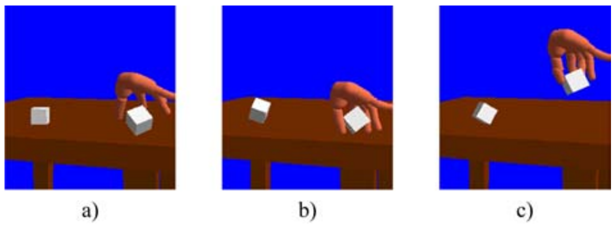

Fig. 6 Grasping a box with 12 polygons at 250 frames per second

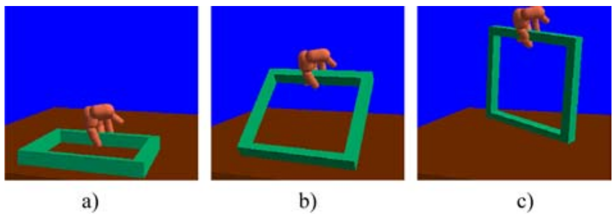

Fig. 7 Grasping a frame with 38 polygons at 250 frames per second

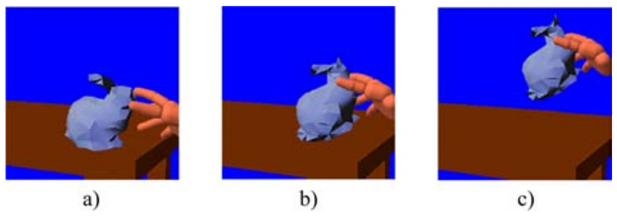

Fig. 8 Grasping the Stanford bunny with 250 polygons at 230 frames per second

loop is responsible for the device refreshing, the constraint force computation of the hand model, and updates the object positions and orientations of the visual representation according to the collision and simulation results.

Ve² is implemented in C++ and uses the following software packages: World Toolkit R9 for visualisation and driver for the FASTRAK tracking system, SOLID 3.5 for collision detection, ODE v0.039 as physics engine, the GSL math library version 1.1.1 for matrix and vector computation, and the Virtual Hand Suite 2000 from Immersion is used as device driver for the CyberGrasp. For the system parameters of the presented tests see Tab 1. For the constraint force parameters of the hand model of the finger limbs and palm see Tab. 2. The results of three different grasping tests with the current configuration are shown in the pictures Fig. 6, Fig. 7, and Fig. 8. Due to hardware restrictions of the SGI platform simplified objects were chosen to show the principle of the proposed method.

6. Conclusion

A concept for the integration of generally available software packages in order to support dexterous haptic interaction was proposed. As one representation of a generally available collision detection package SOLID was used, which employs the GJK algorithm. As rigid-body dynamics simulation package ODE was used, which employs the time stepping approach and uses linear complementary techniques to model and simulate multi-rigid-body dynamics with contact and friction. This presented combination of algorithms integrated in Ve² allows a stable and realistic manipulation of dexterous objects in virtual environments. The user is enabled to touch, feel, and manipulate the virtual objects.

Further work will be conducted to port the underlying VR-system Ve² onto Windows and Linux platforms to enhance the overall system performance in order to handle larger objects and to maintain higher update rates for the collision detection and simulation loop. Besides that, we are interested to expand the collision detection algorithm implemented in SOLID in a manner, that not only the maximum penetration depth of two colliding objects is given back, but also the penetration depth of every single polygon. This would enable more elaborate contact calculations, necessary for a stable 6DOF haptic renderer.

Table 1. System Parameters

Integration stepsize *h* [sec]	0.001
Gravity [m/s²]	0.8
Mass *m* per body of all manipulated bodies [kg]	1.0
Intertia tensor **I** per body [kg m²]	Appro. as sphere, depending on bounding box volume
μ [--]	200
ERP [--]	0.01
CFM [m/Ns]	0.01

Table 2. Hand Model Parameters

Mass of limbs *m* [kg]	0.01
Intertia tensor **I** per body [kg m²]	see Tab. 1
k_f [N/m]	200
d_f [N s/m]	800
k_t [N/m]	0.48
d_t [N s/m]	0.06

7. Acknowledgment

The authors would like to thank the Bayerisches Kompetenznetzwerk fuer Mechatronik (BKM), which financed this project. The BKM is an initiative of the High-Tech-Offensive Bayern.

References

[1] W. McNeely, K. Puterbaugh, and J. Troy, “Six degree-of-freedom haptic rendering using voxel sampling.” in *Proc.of ACM SIGGRAPH*, 1999, pp. 401–408.

[2] K. Hirota and M. Hirose, “Dexterous object manipulation based on collision response,” in *Proceedings of the IEEE Virtual Reality 2003*, 2003, pp. 232–239.

[3] M. Reggiani, M. Mazzoli, and S. Caselli, “An experimental evaluation of collision detection packages for robot motion planning,” in *Submitted to IEEE Int. Conf. on Robotics and Automation*. IEEE, 2002.

[4] E. Gilbert, D. Johnson, and S. Keerthi, “A fast procedure for computing the distance between complex objects in three-dimensional space,” *IEEE Transactions on Robotics and Automation*, vol. 4, no. 2, April 1988, pp. 193–203.

[5] S. Cameron, “Enhancing gjk: Computing minimum and penetration distances between convex polyhedra.” in *In Proc. IEEE Int. conf. on Robotics and Automation*, April 1997, pp. 3112–3117.

[6] G. van den Bergen, *Collision Detection in Interactive 3D Environments*, ser. Series in Interactive 3D Technology. Amsterdam, Boston, Heidelberg: Morgan Kaufmann Publishers, 2004.

[7] C. Luciano, P. Banerjee, T. DeFanti, and S. Mehrotra, “Realistic cross-platform haptic applications using freely-available libraries,” in *Proceedings of the* 12th *Symposium on Haptic Interfaces for Virtual Environment and Teleoperator Systems, in conjunction with IEEE Virtual Reality 2004*, 2004.

[8] D. Baraff, “Analytical methods for dynamic simulation of non-penetrating rigid bodies,” in SIGGRAPH *’89*, ser. Computer Graphics, vol. 23, no. 3, 1989.

[9] A. Witkin and D. Baraff, “Physically based modeling: Principles and practice,” *SIGGRAPH 1997 Course Notes*, 1997. [Online]. Available: http://www.cs.cmu.edu/baraff/sigcourse/notesa.pdf (July 2004).

[10] J. M. Brown and J. E. Colgate, “Physics-based approach to haptic display,” in *Proceedings of the 1994 International Symposium on Measurement and Control in Robotics, Topical Workshop on Virtual Reality*, Houston, TX, 1994, pp. 101–106.

[11] Y. Ikei, K. Takahashi, and S. Fukuda, “Statics-based contact behavior simulation of a manipulated object,” in *International Conference on Artificial Reality and Telexistence, ICAT’98*, 1998, pp. 77–82.

[12] S. Hasegawa, N. Okada, J. Baba, Y. Tazaki, H. Ichikawa, A. Shirai, Y. Koike, and M. Sato, “Springhead: Open source haptic software for virtual worlds with dynamics simulations,” in *Proceedings of EuroHaptics 2004 (CDROM)*, 2004.

[13] S. Hasegawa and M. Sato, “Real-time rigid body simulation for haptic interactions based on contact volume of polygonal objects,” *EUROGRAPHICS*, vol. 23, no. 3, 2004.

[14] R. Kijima and M. Hirose, “Fine object manipulation in virtual environment,” in *2nd Eurographics Workshop on Virtual Environment*, 1995, pp. 1–10.

[15] R. Boulic, S. Rezzonico, and D. Thalmann, “Multi-finger manipulation of virtual objects,” in *ACM Symposium on Virtual Reality Software and Technology VRST’96*, Hong-Kong, July 1996, pp. 67–74.

[16] Z. Huang, R. Boulic, and D. Thalmann, “A multi-sensor approach for grasping and 3-D interaction,” in *Computer Graphics International ’95*, June 1995.

[17] R. Furusawa, N. Abe, K. Tanaka, K. Matsunaga, and H. Taki, “Presenting states and functions of objects under assembling operation with force display device,” in *International Conference on Artificial Reality and Telexistence, ICAT’98*, 1998, pp. 175–181.

[18] K. Hirota, M. Hirayama, A. Tanaka, and T. Kaneko, “Representation of force in object manipulation,” in *International Conference on Artificial Reality and Telexistence, ICAT’99*, 1999, pp. 237–243.

[19] R. Smith, *Open Dynamics Engine v0.5 User Guide*. [Online]. Available: http://ode.org/ode-latestuserguide.html (July 2004).

[20] D. Baraff, “Fast contact force computation for nonpenetrating rigid bodies,” in *SIGGRAPH ’94*, ser. Computer Graphics Proceedings, Annual Conference Series, 1994, pp. 23–34.

[21] D. Baraff, “Linear-time dynamics using lagrange multipliers,” in *SIGGRAPH ’96*, ser. Computer Graphics Proceedings, Annual Conference Series, 1996, pp. 137–146.