A Benchmarking Suite for 6-DOF Real Time Collision Response Algorithms

Rene Weller*
Clausthal University, Germany
David Mainzer†
Clausthal University, Germany
Gabriel Zachmann‡
Clausthal University, Germany
Mikel Sagardia§
German Aerospace Center (DLR)
Thomas Hulin¶
German Aerospace Center (DLR)
Carsten Preusche‖
German Aerospace Center (DLR)

Abstract

We present a benchmarking suite for rigid object collision detection and collision response schemes. The proposed benchmarking suite can evaluate both the performance as well as the quality of the collision response. The former is achieved by densely sampling the configuration space of a large number of highly detailed objects; the latter is achieved by a novel methodology that comprises a number of models for certain collision scenarios. With these models, we compare the force and torque signals both in direction and magnitude.

Our device-independent approach allows objective predictions for physically-based simulations as well as 6-DOF haptic rendering scenarios. In the results, we show a comprehensive example application of our benchmarks comparing two quite different algorithms utilizing our proposed benchmarking suite. This proves empirically that our methodology can become a standard evaluation framework.

CR Categories: I.3.6 [Methodology and Techniques]: Computer Graphics—Graphics data structures and data types

Keywords: Benchmarking, Collision Detection, Haptics

1 Introduction

In order to make games or virtual environments realistic, one of the fundamental technologies is collision handling. It detects collisions among virtual objects, computes a collision response (such as penetration depth, contact points, and contact normals), and finally feeds these into a physically-based simulation or force-feedback algorithm.

Especially with forces, human perception is very sensitive to unexpected discontinuities both in magnitude and direction [Kim et al. 2002]. This effect is aggravated particularly, when both visual and haptic feedback is provided to the user: it is known that visual and tactical sensations are treated together in a single attentional mechanism, and wrong attention uses can affect the suspension of disbelief [C. Spence and Driver 2000]. Consequently, collision detection algorithms should provide stable and continuous forces and torques, even in extreme situations like high impact velocities or large contact areas. Moreover, they should provide these forces at interactive rates. For this reason, a standardized benchmark would help users to classify collision handling systems with respect to their special requirements in advance.

In many applications, an additional requirement is that the collision detection must be very fast. In particular, force-feedback requires a constant update rate of 1000Hz. Additionally, penalty-based physical simulations often perform a number of iterations for a single rendering frame, requiring collision detection at $n \times 30$Hz, if the scene is rendered at 30Hz.

Overall, a benchmarking suite for collision detection should not only assess its performance, but also the quality of its collision response.

The benchmarking suite we propose in this paper achieves both of these goals. Proposed tests are very simple and can be downloaded from our website[1]. Thus, it should be very easy for developers to select the algorithm best suited to their needs; in addition, it should be possible for researchers to assess not only the performance but also the quality of new collision detection algorithms.

1.1 Our Approach

Like most collision handling systems, our benchmark is divided into two separate parts: The kinematic problem of collision detection is primarily investigated with respect to performance, whereas the dynamic problem of collision response corresponds to the quality of the forces and torques.

In order to test the performance, our collision detection benchmark covers a wide variety of different, highly detailed objects in a vast number of different configurations, including situations of close proximity without contact as well as situations reaching from light to heavy interpenetrations. Configurations with penetrations are important because most simulations are penalty-based and thus, computed forces are related to the intersection (amount).

In order to determine the collision response quality of an algorithm, we pursue a different approach, because computing realistic forces and torques from detailed objects in complex contact scenarios is highly non-trivial.

Because of that, we propose to use fairly simple scenarios and geometry tests to measure the quality of the collision response. We believe that this approach is even more warranted because different collision handling systems use different measures for the force and torque computations. For instance, penalty-based methods usually use a translational penetration depth or the penetration volume, impulse based collision response schemes often need the first time of impact.

Another advantage of simple scenarios is that we can model them, which allows us to calculate the theoretically expected forces and torques analytically for different collision response schemes. The comparison of this analytically derived ground truth data with the

*e-mail: rwe@tu-clausthal.de
†e-mail: dm@tu-clausthal.de
‡e-mail: zach@tu-clausthal.de
§e-mail: mikel.sagardia@dlr.de
¶e-mail: thomas.hulin@dlr.de
‖e-mail: carsten.preusche@dlr.de

[1] http://cg.in.tu-clausthal.de/research/colldet_benchmark/index.shtml

data gathered from the benchmarked algorithms allows us to define several measures, such as deviations and discontinuities of forces and torques, or the measurement of noise.

Our benchmarking suite contains several artificial scenes that support different challenges for collision handling schemes, including scenarios with thin sheets and large contact areas.

Summarizing, our benchmarking suite proposed in this work contributes with:

- a performance benchmark for collision detection algorithms;
- an evaluation method for force and torque quality that analyzes both magnitude and direction values with respect to contact models;
- a validation of our proposed benchmark;
- and a thorough evaluation of two rather different collision detection algorithms.

This last point empirically proves that our methodology can become a standard evaluation framework. The combination of both performance and quality benchmarks allows for the identification of specific strengths and weaknesses and, thus, a realistic rating of each benchmarked algorithm. Moreover, our benchmark helps to identify specific scenarios where an algorithm's collision response diverges from the expected results.

2 Previous Work

A first approach to a comprehensive and objective benchmarking suite was given by [Zachmann 1998]. The code for that benchmark is freely available. However, it cannot produce configurations with a predefined distance or penetration. This is problematic because in many simulations, objects slide along each other or penetrate only slightly.

Many collision detection researchers design their own benchmarks. For example, [Cohen et al. 1995] measured the performance of their collision detection algorithm by using a multi-body simulation as a benchmark. Van den Bergen [Van Den Bergen 2005] used three static benchmarks where a pair of models — a torus, teapot, and an X-wing — were placed randomly in a bounded space and tested for intersection. The probability that the two objects would collide was set to approximately 60%. Govindaraju [Govindaraju et al. 2005] created a benchmark for deformable bodies. Other researchers have focused on benchmarking of physics engines, in which collision detection play an essential role. The Physics Abstract Layer (PAL) [Boeing and Bräunl 2007] provides with a unified and solid interface to physics engines. Using PAL, a set of benchmarks has been constructed. The collision detection benchmark simulates sixty-four spheres falling into an inverted square pyramid. The downside of this benchmark is that it is a very special scenario. Caselli et al [Caselli et al. 2002] evaluate several recent collision detection libraries within the context of motion planning for rigid and articulated robots in 3D workspaces. But this benchmark is not of general utility and is restricted to a fixed set of scenarios.

Cao presents a framework for benchmarking haptic systems [Cao 2006]. This framework emulates a 3-DOF point-based haptic device, to which benchmarks can be attached. Another problem is that it is unsuitable for benchmarking non-haptic algorithm behavior. [Ruffaldi et al. 2006] presents a series of ground truth data sets for haptic rendering. These data can be used to assess the accuracy of a particular haptic rendering system, but this benchmark only approximates a single point of contact.

3 Description of the Benchmarks

Our benchmarking suite consists of a Performance Benchmark and a Force and Torque Quality Benchmark. Thanks to them we can compare the collision detection time and the computed force and torque.

3.1 Performance Benchmark

The Performance Benchmark has two scenarios. **Scenario I** simulates situations where objects are in close proximity, but not touching, while **Scenario II** simulates situations where two objects intersect. The relative position of both objects is given by a *configuration*. A configuration consists of 6 parameters shown in Figure 1: the transformation of object B in the coordinate system of object A, given by d, φ_A, θ_A and the rotation of object B, given by φ_B, θ_B, ψ_B.

Most collision detection libraries for static collision detection between rigid objects are based on bounding volume hierarchies (BVHs). The worst case for these BVH-based algorithms is any situation in which bounding leaves collide even in deep levels in the hierarchy, but actually no polygon collision occurs.

For a pair of objects with a given shape, the most relevant parameters that determine collision detection time are the relative position, orientation, distance, and, to a lesser amount, their complexity: for BVH-based approaches. Since we cannot foresee the application of a given collision detection algorithm, the relative position and orientation are more or less random, from a statistical point of view. Therefore, it seems reasonable to factor these parameters out. We achieve this for Scenario I by testing as many configurations as possible for a given distance and complexity. For Scenario II, we fix the intersection volume and test as many configurations as possible. Thus, for a given geometry and a given algorithm, we obtain an average collision detection time as a foundation of the separation distance or the intersection volume, respectively.

In order to generate vast numbers of configurations we used a modified version of the procedure proposed in [Trenkel et al. 2007]. Libraries for the collision detection algorithm described in Section 4.1 were included. In addition, it is now possible to compute configurations for Scenario II, i.e., configurations where objects intersect with a predefined intersection volume. Because of the high computation time (see Section 3.1.1) we implemented the functionality to run the benchmark on a cluster.

3.1.1 Computation of configurations

In order to calculate the distance d between two objects we need the two closest points from object A and object B. So as to be scale-invariant, the distance is given in percent of the whole bounding box of object A.

Object A has a fixed position and object B is placed on a sphere around object A. This sphere must be bigger than the bounding box of object A plus the given distance d between both objects. In the next step, we move object B on a straight line to the center of object A until we reach the required distance or intersection volume. This configuration is then stored.

Our search space has 6 dimensions. To get as many configurations as possible consequently the configuration space must be sampled densely. For Scenario I we chose a step size of 15° for the spherical coordinates and a step size of 15° per axis for the rotations of object B. With these values, we generated a set of 1 991 808 sample configurations for each distance.

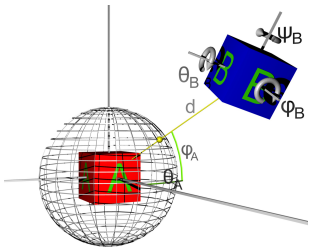

Figure 1: *The search space that is sampled by our sphere-method.*

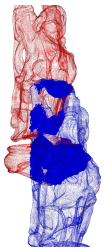

Figure 2: *Happy buddha with 10% intersection volume*

We computed sample configurations for distances from 0 up to 30% of the object size in 1% steps, because in all example cases, there was no significant time spent on collision detection for larger distances. To compute the configuration of two objects with the correct distance we used PQP [Gottschalk et al. 1996; Larsen et al. 1999].

For Scenario II we used Inner Sphere Trees (IST) (see Section 4.1.2) to compute the configurations. A tetrahedron-based approach could not be used because of the probability relatively large intersection times. Although ISTs compute intersection volumes very quickly, we still had to reduce the sampling of the configuration space. Therefore, we changed the step size per axis to 30°. We computed sample configurations for intersection volumes from 0 up to 10% of the total fixed object volume in 1% steps. With these values, we generated a set of 268 128 sample configurations for every intersection volume. Because most applications of collision detection try to avoid collision/intersection, an intersection volume of 10% seems more than enough, as shown in Figure 2.

To compute all these configurations we used a PC cluster with 25 cluster nodes, each with 4 Intel Xeon CPUs and 16GB of RAM. The time needed to calculate configurations for a complete set of distances or intersection volumes varies from object to object between 10h and 200h. Overall, we computed configurations for 86 objects, which lasted 5 600 CPU days.

3.1.2 Benchmarking

Benchmarking is not as time consuming as configuration computation. To perform the benchmark, we load the set of configurations for one object. For each object-object distance and intersection volume respectively, we start timing, set the transformation matrix of the moving object to all the configurations associated with that distance, and perform a collision test for each of them. After that, we get a maximum- and an average collision detection time for the given distance or intersection volume, respectively. Overall, we did 65 million different collision detection tests with each collision library.

3.2 Force and Torque Quality Benchmark

The quality benchmark evaluates the deviation of the magnitude and direction of the virtual forces and torques ideal prediction models. Ideal forces and torques will be denoted by $\mathbf{F}^{\mathrm{i}}$ and $\mathbf{T}^{\mathrm{i}}$, respectively, while the ones computed by one of the collision detection algorithms — measured forces — will be denoted by $\mathbf{F}^{\mathrm{m}}$ and $\mathbf{T}^{\mathrm{m}}$.

Consequently, the scenarios in this benchmark, including objects and paths, should be meet two requirements: a) they should be simple enough so that we can provide a model; and b) they should be a suitable abstraction of the most common contact configurations in force feedback or physically-based simulations.

This Section introduces the implemented scenarios (Section 3.2.1) and methodology (Sections 3.2.2 and 3.2.3) in order to evaluate force and torque quality.

3.2.1 Benchmarking Scenarios

Figure 3 shows all scenarios with their parameters; they are explained in the following.

Scenario I (a,b): A cone is translated while colliding with a block, maintaining a constant penetration. The penetration we chose is $\delta = \frac{1}{3}H = \frac{2}{3}r$ and the length of the trajectory is $L + 2a$. Two situations have been differentiated in this scenario: (a) $h > \delta$ and (b) $h \to 0$, i.e., the block is a *thin* rectangle.

Ideally, only forces should appear and they should have only a component in the positive y direction. Moreover, these forces should be constant while the cone slides on the block. This scenario evaluates the behavior of algorithms with objects that have flat surfaces or sharp corners. In addition, Scenario Ib evaluates how algorithms handle the so-called *tunneling effect* which occurs when thin or non-watertight objects yield too small forces and torques that allow interpenetration.

Scenario II: A sphere is revolved around a cylinder maintaining a constant penetration. The radius of the orbit is $\rho = \frac{5}{3}R = \frac{5}{3}r$. Ideally, only forces should appear (no torques) and they should have uniquely sinusoid components in x and y directions. In addition to that, the measured force magnitude should be constant while the sphere revolves around the cylinder. This is a suitable benchmark for environments with objects that have smooth, rounded surfaces.

Scenario III: A so-called *pins* object with a rectangular and a circular pin and a matching *holes* object compose this scenario. The rectangular pin is introduced in the rectangular hole and is turned around its axis. The size of the objects is $b = 2a$, the side of the rectangular pin is $c = 2r$ and it has a length of a in z direction. The maximum rotation angle is $\phi_{\max} = 30°$. Ideally, only torques should appear and they should have only a component in positive z direction. Moreover, the measured torque magnitude should increase as ϕ increases. This scenario evaluates the behavior of algorithms with large contact areas.

Scenario IV: This scenario uses the same objects as in Scenario III. The start configuration is shown in Figure 3. Then, the pins object is revolved around the central axis of the second one. The orbit radius is $\rho = \frac{1}{10}c = \frac{1}{20}r$. The expected forces and torques are those that bring the *pins* object towards the central axis, i.e., sinusoidal forces on the xy plane and torques with only z component. This scenario evaluates the behavior of algorithms with large and *superfluous* contact areas that should not generate collision reactions, such as the contact between objects in the xy plane. Besides of that, this scenario contains small displacements around a configuration in which two objects are in surface contact. These small displacements should generate the corresponding small forces that push the *pins* object back to the *only-surface-contact* configuration.

3.2.2 Evaluation Method

For each scenario, we measured and recorded the following values for each time stamp k.

1. forces $\mathbf{F}^{\mathrm{m}}_k$,
2. torques $\mathbf{T}^{\mathrm{m}}_k$,
3. penalty values q^{m}_k and

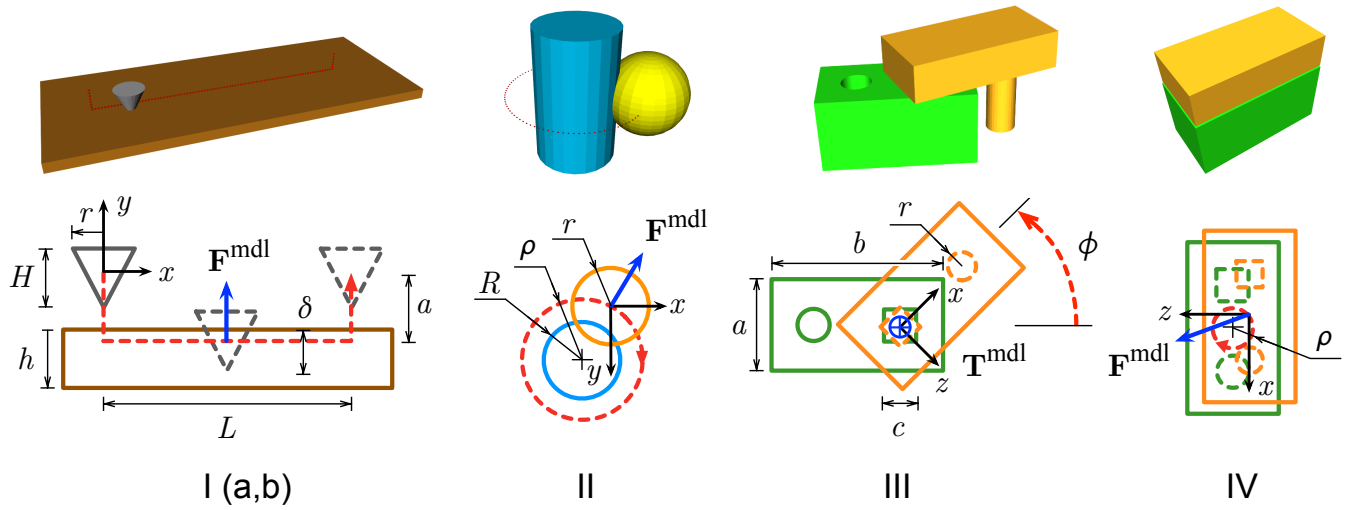

Figure 3: *Scenarios in the force and torque quality benchmark, explained in Section 3.2.1. Upper row shows 3D snapshots, whereas the lower displays parametrized schematics. Trajectories are represented with red dashed curves. Expected relevant forces and/or torques are shown with blue vectors. Coordinate systems are placed in points where forces and torques are measured – for the cone and the sphere this point is in their AABB center, whereas the position in z axis for the pins object is in the middle of the pin.*

4. computation time t_k.

In order to assess these measured values, we have developed ideal models of the expected forces and torques (i). The directions of these force and torque vector models are displayed in Figure 3, whereas magnitudes are considered to be proportional to analytically derivable collision properties, such as

1. $\|\mathbf{F}^{\mathrm{i}}\|$ or $\|\mathbf{T}^{\mathrm{i}}\| \sim p$, translational penetration depth,
2. $\|\mathbf{F}^{\mathrm{i}}\|$ or $\|\mathbf{T}^{\mathrm{i}}\| \sim V$, intersection volume.

In each scenario, we have determined p and V, respectively, as follows:

- Scenario Ia: $p \sim \delta$ and $V \sim \delta^3$
- Scenario Ib: $p \sim \delta$
- Scenario II: $p = \rho = \text{const}$ and $V = \text{const}$
- Scenario III: $p \sim \sin(\frac{\phi}{2}) - 1$ and $V \sim (\frac{1}{\tan(\phi)} + \frac{1}{\tan(\frac{\pi}{2}-\phi)})(\sqrt{2}\cos(\frac{\pi}{4}-\phi)-1)^2$
- Scenario IV: $p = \rho = \text{const}$ and $V = c^2 - (c - \rho|\cos\phi|)(c - \rho|\sin\phi|) + \pi r^2 - 4\int_{\frac{\rho}{2}}^{r}(r^2-\tau^2)d\tau$

In order to evaluate the quality of the magnitude, the standard deviation of measured (m) and ideal (i) curves is computed:

$$\sigma_F = \frac{1}{N}\sqrt{\sum_{k=1}^{N}\left(\,\|\hat{\mathbf{F}}^{\mathrm{i}}_k\| - \|\hat{\mathbf{F}}^{\mathrm{m}}_k\|\,\right)^2}, \tag{1}$$

where $\hat{\mathbf{F}} = \frac{\mathbf{F}}{\|\mathbf{F}\|_{\max}}$, and N being the total amount of time stamps.

Analogously, the indicator for direction deviation is the angle between ideal and measured values; the average value of this angle is:

$$\gamma_F = \frac{1}{N}\sum_{k=1}^{N}\arccos\frac{\mathbf{F}^{\mathrm{i}}_k\mathbf{F}^{\mathrm{m}}_k}{\|\mathbf{F}^{\mathrm{i}}_k\|\|\mathbf{F}^{\mathrm{m}}_k\|}. \tag{2}$$

Deviation values for torques (σ_T, γ_T) are computed using $\mathbf{T}^{\mathrm{m}}_k$ and $\mathbf{T}^{\mathrm{i}}_k$, instead of force values.

Additionally, we measure the amount of noise in the measured signals. A color coded time-frequency diagram using short time Fourier transform can be used to visualize the noise in time domain. In order to define a more manageable value for evaluations, we compute the ratio

$$\nu = \frac{\int S^{\mathrm{m}}}{\int S^{\mathrm{i}}}, \tag{3}$$

where S^{m} is the energy spectral density of the measured variable (e.g. $\|F^{\mathrm{m}}\|$) and S^{i} is the spectrum of the corresponding ideal signal. ν can be evaluated for forces and torques directions and magnitudes separately.

3.2.3 Equivalent Optimized Resolutions for Comparing Different Algorithms

Usually, when increasing the resolution quality is improved, whereas computation time increases. Therefore, an appropriate trade-off between quality and time performance must be found.

When properly evaluating or comparing collision detection algorithms, such a resolution must be found that makes possible to compare algorithms' quality for a given average performance, or to compare their performance for a given desired quality. In this context, we name "equivalent" optimized resolutions such resolutions with which algorithms exhibit a same desired time performance, being possible to fairly compare their qualities.

Considering two objects in a scenario (A is dynamic, B is static), we define the resolution pair $(e^A_{\mathrm{opt}}, e^B_{\mathrm{opt}})$ to be the optimum *equivalent* resolution pair:

$$(e^A_{\mathrm{opt}}, e^B_{\mathrm{opt}}) = \min\{\eta(e^A, e^B) \mid \bar{t}(e^A, e^B) = \tau\}, \tag{4}$$

where τ is the maximum admissible average computation time, $\bar{t}$ and $\eta = \omega_\sigma\sigma + \omega_\gamma\gamma$, the equally weighted sum of the standard deviations.

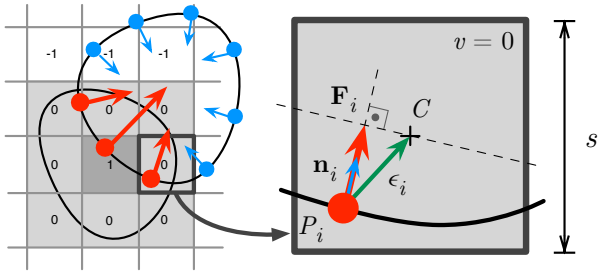

Figure 4: *On the left, a layered voxmap (bottom) is colliding with red pointshell points (top), yielding red bold collision forces. On the right, the computation of a single collision force related to a colliding point is graphically shown. Single collision forces are computed scaling the normal vector ($\mathbf{n}_i$) of the colliding point (P_i) with the sum of the local ($\mathbf{n}_i\varepsilon_i$) and global (vs) penetration of the point in the object.*

In practice, since time and quality functions of Equation 4 are unknown, performed evaluations were carried out numerically after running several tests. For each scenario and algorithm, we defined three different resolutions within a reasonable[2] domain for each object A and B, building sets of $3 \times 3 = 9$ pairs (e^A, e^B). Then, the sets of 9 corresponding tests were performed, recording all necessary average computation times ($\bar{t}$) and global deviations (η) in each one. Next, we applied a linear regression to values of $\bar{t}$, obtaining the plane which predicts the average computation time for a resolution pair in each scenario. Each of these planes was intersected with $\tau = 0.9\text{ms}$[3], obtaining the lines formed by all (e^A, e^B) expected to have $\bar{t} = 0.9\text{ms}$ for each scenario.

Being aware of the fact that further refinements would yet be possible, it is considered that the reached compromise is accurate enough in order to make a fair comparison. The average absolute difference between predicted and measured η values with *equivalent* resolutions was 1.2% for the VPS algorithm and 2.1% for the IST algorithm.

4 Results

In order to test our benchmark, we used two collision detection algorithms, VPS and IST. Both algorithms use a penalty-based haptic rendering method, which allows colliding objects to penetrate each other to some degree. Each algorithm uses different penalty values: the one from VPS is the penetrated *distance*, while the one from IST is the intersection volume.

First, we explain the algorithms and how they compute force and torque values they return to our benchmark. After this, we discuss the output of the Performance Benchmark and the Force and Torque Quality Benchmark.

4.1 The Collision Detection Algorithms

4.1.1 The Voxmap-Pointshell (VPS) Algorithm

The Voxmap-Pointshell (VPS) Algorithm was initially presented by [McNeely et al. 1999]. The algorithm computes collision forces and torques of potentially big and complex geometries with 1kHz update rates. To achieve this goal, two types of data structures are generated offline for each colliding object-pair: a voxmap and a pointshell (see Figure 4). In this work, we used the fast and accurate voxmap generator presented by [Sagardia et al. 2008].

Voxmaps are 3D grids in which each voxel stores a discrete distance value $v \in \mathbb{Z}$ to the surface. Pointshells are sets of points uniformly distributed on the surface of the object; each point has additionally an inwards pointing normal vector.

During collision detection, the normal vectors $\mathbf{n}_i$ of colliding points P_i — those which are in voxels with $v \geq 0$ — are summed, after being weighted by their penetration in the voxmap, yielding the collision force $\mathbf{F}$. Torques $\mathbf{T}_i$ generated by colliding points are the cross product between forces $\mathbf{F}_i$ and point coordinates P_i, all magnitudes expressed in the pointshell frame, with its origin in with being the center of mass. At the end, these torques $\mathbf{T}_i$ are summed to compute the total torque $\mathbf{T}$.

4.1.2 The Inner Sphere Tree (IST) Algorithm

Inner Sphere Trees [Weller and Zachmann 2009] are a novel geometric data structure, that provides hierarchical bounding volumes from the *inside* of an object. The main idea is to fill the interior of the model with a set of non overlapping spheres that approximate the object's volume closely. Therefore ISTs and, consequently, the collision detection algorithm are independent of the geometry complexity; they only depend on the approximation error.

The penetration volume corresponds to the water displacement of the overlapping parts of the objects and, thus, leads to a physically motivated and continuous repulsion force. The algorithm determines all pairs of overlapping spheres and computes a force for each of them. Summing all these pairwise forces gives the total penalty force $\mathbf{F}$. Similarly, the torque is computed separately for each pair of intersecting spheres and accumulated to obtain the total torque $\mathbf{T}$.

4.2 Discussion of the Benchmark Results

In this section we present the results returned from our benchmarks. The algorithms presented in Section 4.1 were used for this propose.

It is very hard to tell which algorithm is better because this is very dependent on the requirements. Our benchmark provides a wide range of test cases to evaluate the given algorithm and return the computation time and the computed values. In the next sections we explain the results returned by the tested algorithm.

4.2.1 Results of the Performance Benchmark

Appart from the distance or the penetration depth between objects, the performance of the most collision detection libraries mainly depends on the complexity and the shape of the objects. Figure 5 shows some of the objects we used. All objects that are in the public domain can be accessed on our website. Within our benchmarks, we tested a model against a copy of itself. Of course, our benchmark also supports the use of two different objects, but the first method is sufficient to draw conclusions about the performance of the libraries.

We tested the libraries on an Intel Core2 CPU 6700 @ 2.66GHz and 2GB of RAM running Linux. All source code was compiled with gcc 4.3.

An example of a result of the Performance Benchmark is shown in Figure 6, using Happy Buddha as object. Our Performance Benchmark facilitates a comparison of different algorithms as well as an

[2] Between coarse but acceptable and too fine resolutions.

[3] Collision detection and force computation must lie under 1ms; hence, we chose a resonable value under this barrier.

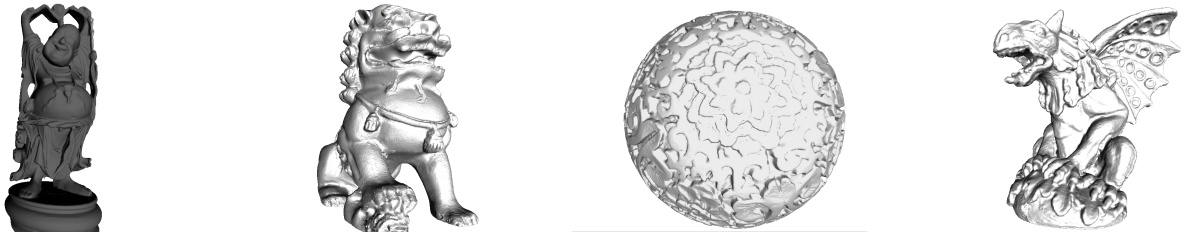

Figure 5: *Some of the objects we used in our Performance Benchmark: A model of a Happy Buddha (1 087 716 polygons), a Chinese Dragon (1 311 956 polygons), a Circular Box (1 402 640 polygons) and a Gargoyle (1 726 420 polygons).*

assessment of the behavior of one algorithm, with respect to the objects complexity.

With the results from the Performance Benchmark it is now possible to compare collision libraries regarding their collision response time. These tests can also be used to determine objects or a placement of two objects which are not ideal for the tested algorithm. It is also possible to determin the influence of the object complexity or a lower approximation error on the collision response time.

However, the computation time is not enough to fully assess a collision detection algorithm. Often, the quality of the collision responses is another important factor. This is discussed in the next section.

4.2.2 Results of the Force and Torque Quality Benchmark

As in the case of the Performance Benchmark, all objects and paths used in the Force and Torque Quality Benchmark (see Figure 3) are available on our website[4]. We tested them on an Intel Core2Quad CPU Q9450 @ 2.66GHz and 3.4GB of RAM running Linux SLED 11. The libraries were compiled with gcc 4.3.

For the voxel size s we have chosen a fixed lenght unit u in the voxelized objects such that $H = 60u, h = 30u$ (Scenario I), $R = 30u$ (a penetration of $20u$ is maintained) (Scenario II), $c = 20u$ (Scenario III), and $\rho = 20u$ (Scenario IV). The number of voxels was chosen to be $728 \times 24 \times 303$ voxels for the block in Senario I while the cone has 15669 pointshell points. In Scenario II, we used $491 \times 816 \times 491$ voxels for the cylinder and 12640 pointshell points for the sphere. In Scenario III, the number of voxels was chosen to be $1204 \times 604 \times 603$ for the holes and 12474 pointshell points for the pins object. For the last Scenario, the number of voxels was chosen to be $243 \times 123 \times 123$ for the holes and 13295 pointshell points for the pins object.

Figures 7 and 8 show example plots of the magnitude analysis. The left side of Figure 7 contains the expected model curves for ideal force magnitudes in Scenario I. Measured curves are superposed to expected curves to give an idea of how reliable they are derived with respect to these proposed collision response models. The standard deviation between measured and ideal curves yields the magnitude deviation $\sigma_F = 0.043$ for VPS and $\sigma_F = 0.176$ for ISTs. In Scenario III, the standard deviation between measured and ideal curves yields the magnitude deviation $\sigma_T = 0.169$ and $\sigma_T = 0.112$ for the torques, respectively. The right side of Figure 7 shows the curve $\frac{\|T\|}{\|F\|}$, which should be 0 for Scenario II, since ideally no torques should appear. This quotient gives information about the magnitude of forces or torques that actually should not occur.

In Figure 9, force and torque components are displayed, giving a visual idea of force and torque direction deviations. The left plot of Figure 8 shows this direction deviation for Scenario II, the associated γ values are $\gamma_F = 2.40$ for VPS and $\gamma_F = 7.64$ for ISTs.

Finally, Figure 10 shows the results of our noise measurement of the force in the x-direction in Scenario III. The color coded time-frequency diagrams visualize the amount, the time, and the frequency of the signal's noise. The corresponding ν values are $\nu_F = 0.620$ for VPS and $\nu_F = 1.12$ for ISTs, where values closer to one denote a minor amount of noise.

All these results show that VPS and IST are very close to their underlying models and that different haptic rendering algorithms can be evaluated. From these results we can say that our models for penetration are suitable. Furthermore, they prove empirically that our benchmark is vaild. In particular, the benchmark also reveals significant differences between the algorithms: Whereas ISTs seem to have a higher standard deviation from the ideal model, VPS tends to deliver noisier signal quality. The decision between accuracy and noise could be essential for some applications.

5 Conclusions and Future Work

The results maintain the validity of our analytically derived force and torque models. In addition, they show that quite different collision detection algorithms can be easily benchmarked with our proposed methods.

Our benchmark will be published as open source, so it will be a great asset to users who want to figure out the best suited collision handling scheme to meet their specific requirements, as well as to researchers who want to compare their algorithms with other approaches using a standardized benchmark that delivers verifiable results. Moreover, it helps to identify geometric cases in which the collision response scheme diverges from the correct results.

In the future, it would be nice to generate a ranking of the different measurements, like continuity of forces and torques in magnitude and direction or the noise of the signals, with respect to psychophysical cognition. To achieve that, elaborate user studies need to be done, including testbeds with different haptic devices and investigations about the perception of the different parameters.

Another promising future project would be to extend our benchmarking suite for multi-body-simulations. Finally, a standardized benchmarking suite for deformable objects is still missing and would be very helpful for users and researchers.

Acknowledgment

We would like to thank Ralf Rabätje with Volkswagen AG for the fruitful cooperation. This work was partially supported by DFG grant ZA292/1-1 and BMBF grant Avilus / 01 IM 08 001 U.

[4] http://cg.in.tu-clausthal.de/research/colldet_benchmark/index.shtml

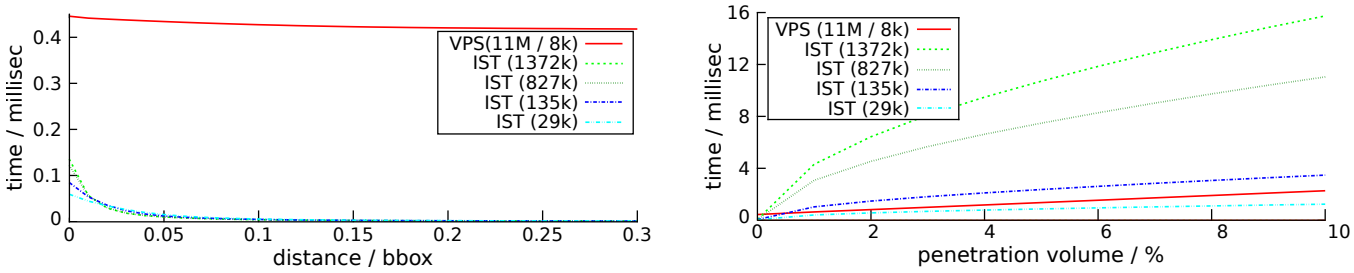

Figure 6: *Performance Benchmark: Example result for Happy Buddha. The left plot shows the measured average collision response time for Scenario I (no collision) and the right one for Scenario II (collision) (see Section 3.1). Distance* 0.0 *means that the objects are touching. Volume 1% means that the intersection volume is equal to 1% of the total object volume. The number in parentheses after IST denotes the number of spheres (see Section 4.1.2). The two numbers after VPS denote the number of voxels and points, respectively (see Section 4.1.1).*

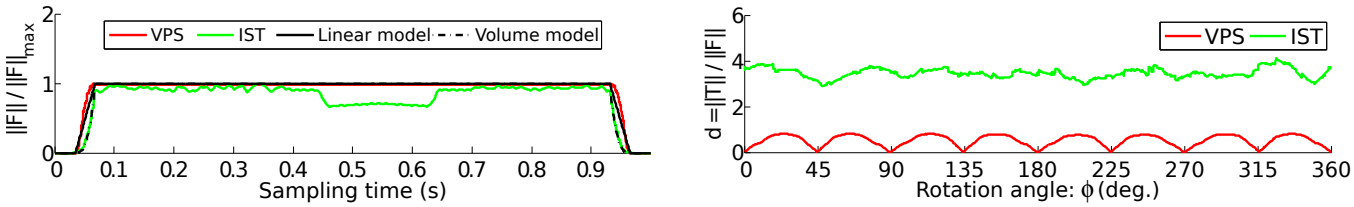

Figure 7: *Force and Torque Quality Benchmark: On the left, an example for the normalized collision force vector computed by the tested algorithms (Scenario I) and on the right the orientation of the vectors (Scenario II).*

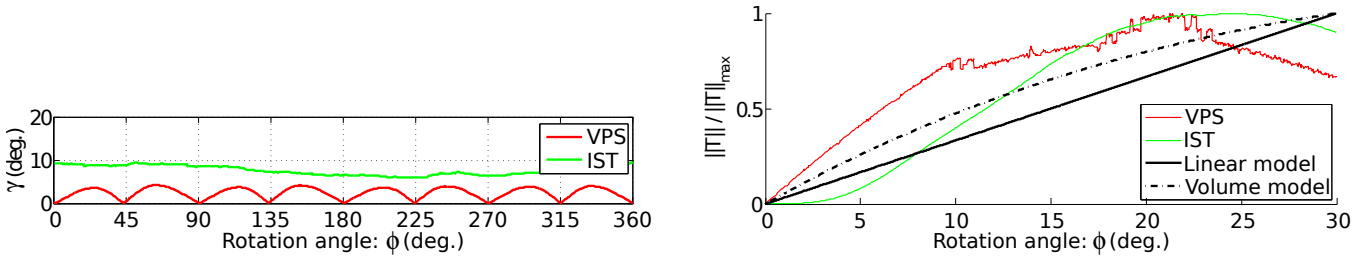

Figure 8: *Force and Torque Quality Benchmark: On the left, an example for an average angle between model and measured forces (Scenario II) and on the right the normalized collision torque vector computed by the tested algorithms (Scenario III).*

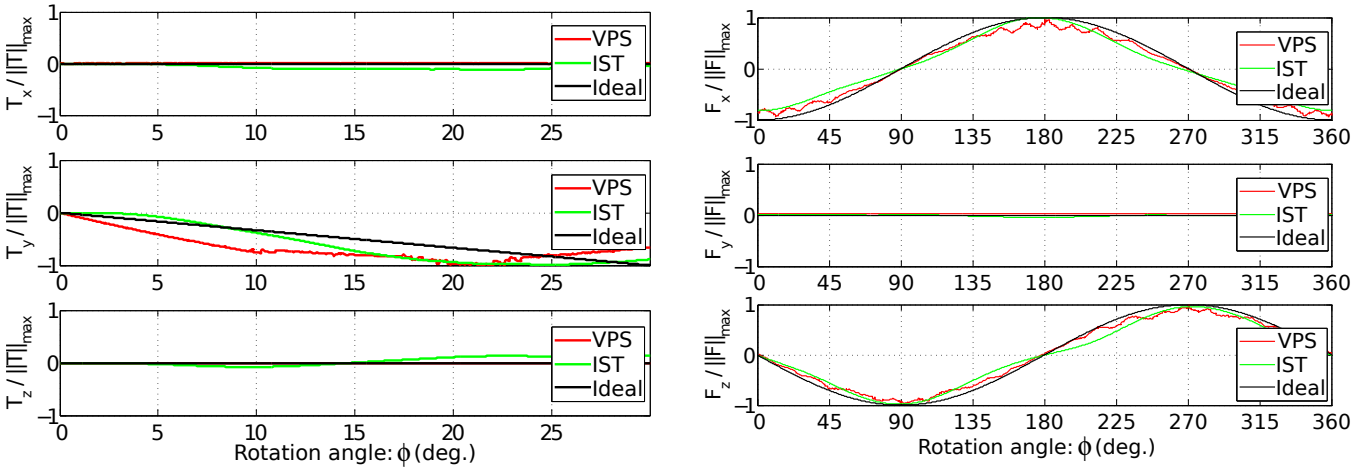

Figure 9: *Force and Torque Quality Benchmark: On the left, an example for the collision torque (Scenario III) and on the right an example for the collision force computed by the tested algorithms (Scenario IV).*

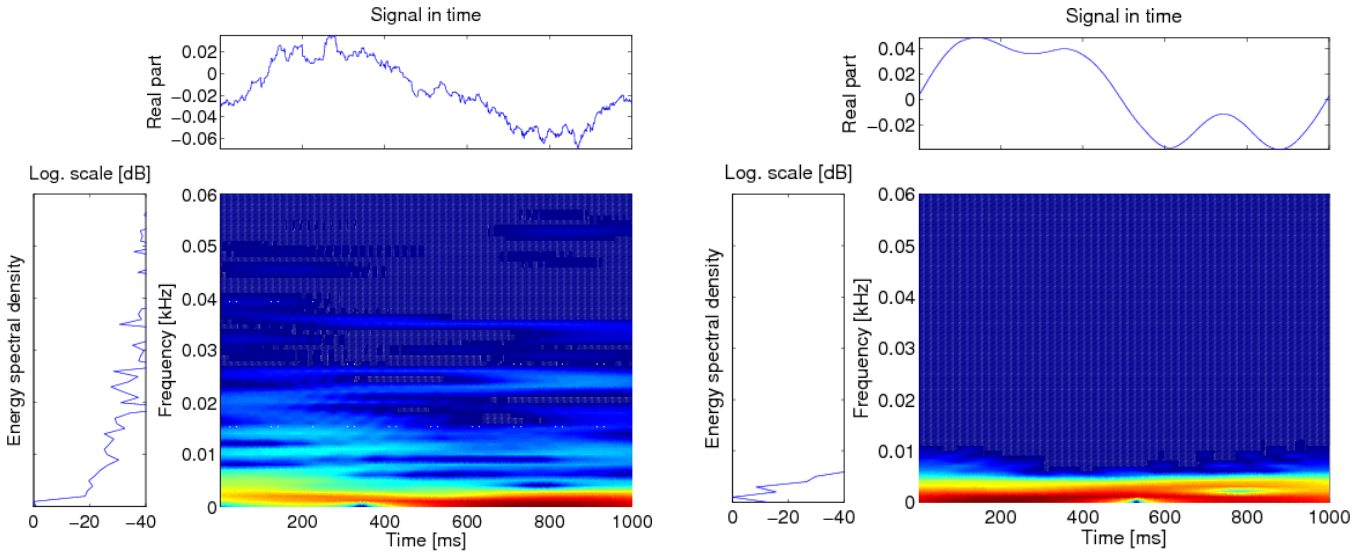

Figure 10: *Force and Torque Quality Benchmark: On the left, the noise in the force signal of the VPS algorithm and on the right noise in force signal of IST algorithm. The colored picture shows the time frequency domain: The colors decode the intensity of the frequency, where dark blue represents an intensity of zero.*

References

BOEING, A., AND BRÄUNL, T. 2007. Evaluation of real-time physics simulation systems. In *Proceedings of the 5th international conference on Computer graphics and interactive techniques in Australia and Southeast Asia*, ACM, 288.

C. SPENCE, F. P., AND DRIVER, J. 2000. Crossmodal links in spatial attention between vision and touch: Allocentric coding revealed by crossing the hands. In *Journal of experimental psychology. Human perception and performance*, 1298–1319.

CAO, X. R. 2006. *A framework for benchmarking haptic systems*. PhD thesis, SIMON FRASER UNIVERSITY.

CASELLI, S., REGGIANI, M., MAZZOLI, M., AND DI PARMA, U. 2002. Exploiting advanced collision detection libraries in a probabilistic motion planner. In *Journal of WSCG–Proceedings of the 10th International Conference in Central Europe on Computer Graphics, Visualization and Computer Vision (WSCG 2002)*, vol. 10, Citeseer, 1–3.

COHEN, J., LIN, M., MANOCHA, D., AND PONAMGI, M. 1995. I-COLLIDE: An interactive and exact collision detection system for large-scale environments. In *Proceedings of the 1995 symposium on Interactive 3D graphics*, ACM.

GOTTSCHALK, S., LIN, M. C., AND MANOCHA, D. 1996. OBBTree: A Hierarchical Structure for Rapid Interference Detection. *Computer Graphics 30*, Annual Conference Series, 171–180.

GOVINDARAJU, N., KNOTT, D., JAIN, N., KABUL, I., TAMSTORF, R., GAYLE, R., LIN, M., AND MANOCHA, D. 2005. Interactive collision detection between deformable models using chromatic decomposition. In *ACM SIGGRAPH 2005 Papers*, ACM, 999.

KIM, L., KYRIKOU, A., DESBRUN, M., AND SUKHATME, G. S. 2002. An implicit-based haptic rendering technique. In *IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2943–2948.

LARSEN, E., GOTTSCHALK, S., LIN, M. C., AND MANOCHA, D. 1999. Fast Proximity Queries with Swept Sphere Volumes. Tech. rep.

MCNEELY, W. A., PUTERBAUGH, K. D., AND TROY, J. J. 1999. Six degree-of-freedom haptic rendering using voxel sampling. In *SIGGRAPH '99: Proceedings of the 26th annual conference on Computer graphics and interactive techniques*, ACM Press/Addison-Wesley Publishing Co., New York, NY, USA, 401–408.

RUFFALDI, E., MORRIS, D., EDMUNDS, T., BARBAGLI, F., AND PAI, D. 2006. Standardized evaluation of haptic rendering systems. *Haptic Interfaces for Virtual Environment and Teleoperator Systems, IEEE VR*.

SAGARDIA, M., HULIN, T., PREUSCHE, C., AND HIRZINGER, G. 2008. Improvements of the Voxmap-PointShell Algorithm-Fast Generation of Haptic Data-Structures. In *53rd IWK-Internationales Wissenschaftliches Kolloquium, Ilmenau, Germany*.

TRENKEL, S., WELLER, R., AND ZACHMANN, G. 2007. A Benchmarking Suite for Static Collision Detection Algorithms. In *Inter'l Conf. in Central Europe on Computer Graphics, Visualization and Computer Vision (WSCG)*, Union Agency, Plzen, Czech Republic, V. Skala, Ed.

VAN DEN BERGEN, G. 2005. Efficient collision detection of complex deformable models using AABB trees. *Graphics tools: The jgt editors' choice*, 131.

WELLER, R., AND ZACHMANN, G. 2009. A unified approach for physically-based simulations and haptic rendering. In *ACM SIGGRAPH Video Game Proceedings*, ACM Press, New Orleans, LA, USA.

ZACHMANN, G. 1998. Rapid collision detection by dynamically aligned DOP-trees. In *Proc. of IEEE Virtual Reality Annual International Symposium; VRAIS '98*, 90–97.