EG UK Theory and Practice of Computer Graphics (2009)
Wen Tang, John Collomosse (Editors)

An Improved Precise Multi-contact Haptic Visualization

Jan Flasar† and Vít Kovalčík and Jiří Sochor

Faculty of Informatics, Masaryk University, Brno, Czech Republic

Abstract

We present an improved multi-contact haptic visualization method based on the Spatialized Normal Cone Hierarchies (SNCH). Though this approach is not entirely new, we have implemented several improvements in order to significantly increase precision and robustness over the previous method. As a consequence we are able to simulate much harder surfaces and give users the chance to feel smaller features on the surface compared to the original approach. This was achieved mostly by using more precise triangle to triangle distance calculations and a different triangle visibility algorithm. As these computations are expensive, we have also developed a new technique to reduce the number of calculations required. Currently, our algorithm is capable of visualizing haptic interactions between two 3D models consisting of tens of thousands of triangles. The simulation is performed in real-time and is seamlessly integrated into a virtual-reality component-based system named VRECKO.

Categories and Subject Descriptors (according to ACM CCS): H.5.1 [Information Systems]: Information Interfaces and PresentationHaptic I/O; I.3.7 [Computing Methodologies]: Computer GraphicsThree-Dimensional Graphics and Realism;

1. Introduction

In the recent decades, we have seen dramatic improvements in the field of computer graphics. In this respect, the quality of today's applications and particularly computer games is very high even when running on a common personal computer. However, current common applications use only visual and aural feedback, while the other senses of users are not stimulated.

In the last years, the number of computer devices with force feedback also increased significantly. With recent technological advances, the time for wider spread of haptic feedback devices is coming. However, the corresponding algorithms are still under development and higher-level simulations are hard to be performed on a common-grade computer.

When solving the problems of haptic interaction with complex structures, we looked for advanced techniques for calculating dynamic responses of moving objects. Therefore, we focused our research on the area of multi-contact collisions. The demands for speed are critical compared to visual rendering, because of the nature of the human sense of touch. While it is enough to render 30 video frames per second to persuade human eyes and brain that an object is

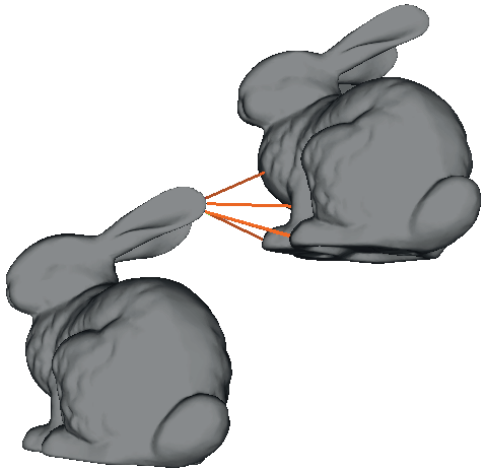

Figure 1: *A set of the closest pairs used to detect forthcoming multi-contact collision between two complex models.*

† e-mail: {flasar | kovalcik | sochor}@fi.muni.cz

DOI: 10.2312/LocalChapterEvents/TPCG/TPCG09/001-008

moving smoothly, the update rate of the haptic feedback simulations has to be significantly higher, with 1000 Hz being the lowest acceptable frequency to achieve smooth feeling of contact response.

Considering single-contact collisions the situation is relatively simple: For a haptic exploration/haptic rendering, we approximate the user's hand with a single point or a very simple object, such as a sphere, and calculate its penetration into other object(s) as the user explores the objects surfaces in the space. With multi-contact collisions, on the other hand, we assume that the user is holding a complex virtual object in their hand and uses it to touch other virtual objects. This requires tracking many different collision points between complex surfaces. To perform this task thousand times per second is possible only using advanced algorithms.

Several different approaches already exist, but each of them has its drawbacks. We have chosen an algorithm satisfying the most of our requirements and modified it to remove some of its limitations.

2. Related Work

Methods for multi-contact collision detection for haptic rendering can be divided into two categories according to the way the force is applied: After (virtual) penetration into an object or before the actual (virtual) collision. *Penetration-based* methods apply the force after the virtual collision is detected, while *proximity-based* methods predict forthcoming collision by some threshold proximity and apply the force before actual collision.

The penetration-based methods are widely used since the beginning of haptic rendering [Bar96]. When a virtual object controlled by a haptic device intersects another object within the scene, the repulsive force is applied. Otaduy et al. [OL03] presented the method that uses contact levels of detail (CLODs) for multiresolution collision queries. It is based on a dual hierarchy created for each object. It includes multiresolution representation and a bounding volume hierarchy. If the algorithm detects penetration into an object then the according force is calculated and applied. For each potential contact it uses error metrics (eg. object-space errors, velocity dependent gap) to select appropriate LOD thus decreasing the computation time.

Kim et al. [KOLM02] developed another 6 DOF haptic rendering method that uses localized contact computations. At first, each non-convex object is decomposed into convex parts and the bounding volume hierarchy is created. If a collision occurs, the penetration depth is calculated using an incremental technique that is based on walking on the surface of the Minkowski difference. The penetration depth is then estimated between overlapping pieces. This method reduces the computational cost and improves the stability of the force applied.

Another novel method was presented in the papers by Barbic et al. [BJ07, BJ08]. It approximates contact forces within a very fast update function and is based on a spatially and temporally adaptive sample-based approach. This method can be also used for haptic visualization of a rigid and a reduced deformable high-poly model. All calculations are performed using a multiresolution point-based representation of one object and a signed-distance field of the other object.

Proximity-based methods applying the forces before the actual penetration into the object, were described in several papers by Johnson et al. [JW03, JW04, JWC05]. Their approach is based on the Spatialized Normal Cone Hierarchies (SNCH) originally presented in [JC01]. SNCH data structure is formed in preprocessing phase for every object participating in the haptic rendering. It is used to set up and maintain the set of the closest pairs of surface features for two objects. The aim is to keep the potentially colliding objects away by generating a repulsive force before they really collide. This is especially useful in various CAD/CAM simulations, because the resulting object path is then guaranteed to be intersection-free. Our solution is based on this approach, therefore additional details are described later in this paper.

3. Multicontact Detection with Spatialized Normal Cone Hierarchies

The original approach by Johnson et al., which serves as a basis for our method, is reasonably fast and attains update rates usually exceeding 1 kHz. However, to do so it only approximates distances between two models by omitting full triangle-triangle distance calculations and using simplified distance instead. This also results in possibly incorrect direction between two triangles, which may cause complete elimination of such triangle pair in following tests. These missing pairs may result in problems in force calculations afterwards. The described problems are infrequent when using 3D models with a dense triangle mesh, but become apparent with models containing large triangles. The approximation also limits the possibilities in the force calculations. It can imitate only "soft" feelings, hard contacts are not present in the simulation.

We have tried to remedy this disadvantage while keeping the speed as high as possible to preserve interactivity. Our method also retains the relatively simple preprocessing and the ability to work with movable objects (though the triangle mesh composing an object should be static to avoid hierarchy recalculation). Another benefit of our method is also the capability of calculating intersection polylines and better haptic operation with models with holes.

3.1. Spatialized Normal Cone Hierarchies

This type of hierarchy is fully described in several papers, most notably [JC01], so we will only sketch the principles here.

The structure consists of a common bounding sphere hierarchy, built upon triangles composing a given object. Every leaf corresponds to a single triangle encapsulated in its bounding sphere.

In each node of the hierarchy, an oriented cone is stored. Each cone represents a range of normals belonging to the surface found in the subtree of the particular node. The axis of the cone is created by averaging the corresponding normals. This is depicted in Figure 2.

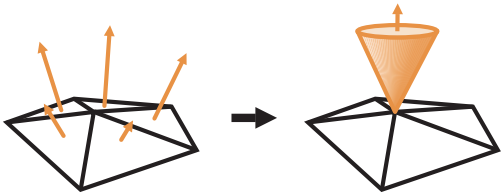

Figure 2: *Normal cone creation.*

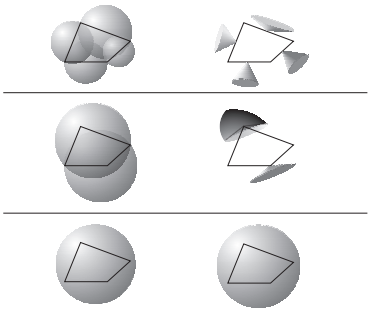

Figure 3: *A 2D example of a SNCH hierarchy for a simple object. All three levels of the hierarchy are shown with bounding spheres on the left and normal cones on the right.*

The surface is stored as a mesh, i.e. for exploration of a triangle's neighbourhood we use connectivity information of triangles and faces, that are connected to a given vertex. Figure 3 shows an example of the SNCH structure.

3.2. Original Algorithm Overview

This section presents an overview of the original algorithm by Johnson et al. Our modifications and differences from this algorithm will be described later.

For estimating multicontact collision between two solids we need to find a set of the closest feature pairs on both surfaces. Using some proximity threshold we may find many close triangle pairs. However, in real situations we need to select some minimal representatives for locally similar feature pairs, to calculate repulsive force properly. A set of such pairs is called a Local Minimum Distance set (LMD). The algorithm creates and maintains this set which is then used to calculate the force.

The SNCH-based multicontact algorithm performs two tasks in parallel: The construction of LMD set by global searching for the LMD pairs and the local update of LMD set.

As a basic structure for rapid detection of collision, the above mentioned SNCH is used.

The latter task updates positions of the LMD pairs in LMD set to keep up with the user movement and is performed in each haptic loop. The global search runs in background to search for a new set of LMD candidates. It uses the SNCH structures to precalculate LMD pairs. Calculating these distances "from the scratch" is computationally expensive, so another thread is used to update LMD set, while the objects are moving. Results from this thread are used to calculate the real force that will be sent to the haptic device.

3.2.1. Global Search

In the description of algorithm, we will use the following notation: A and B are the SNCH structures of the two objects and Φ_A and Φ_B are the currently tested nodes. Each node contains a bounding sphere with centre S_{Φ_A} and radius ρ_{Φ_A}. Also, in each node a cone of normals is stored with axis $\vec{C}_{\Phi_A}$ and half spread angle ϕ_{Φ_A}.

The global search recursively tests nodes from the two SNC hierarchies. It starts by testing the roots against each other. If the pair of nodes passes all the tests, their descendants are also tested (four tests) in the same manner, until the leaves are encountered. The tests performed for each node-node pair are the following, see also Figure 4.

G1. Cutoff pruning based on the spheres distance. If the distance is greater than some preset limit, the nodes are rejected.

G2. Collinearity test (testing cones against each other):

$$\pi - \arccos(\vec{C}_{\Phi_A} \cdot \vec{C}_{\Phi_B}) > \phi_{\Phi_A} + \phi_{\Phi_B}$$

If this condition is true, no pair of collinear vectors from both cones exist, therefore the nodes are rejected.

G3. Test of normal cones vs. the dual solution line cone. Cones do not cover the same subspace, for details see original paper [JC01]

If the tested nodes are leaves with single triangles, additional work is performed:

G4. Each triangle has 7 feature points: 3 vertices, 3 edge-midpoints and face center. For a tested triangle pair, 7×7 point-to-point distances are calculated, a minimal distance is selected.

G5. The vector with selected distance is verified against the vertex/edge/face normals, depending on the feature points which were the closest.

Finally, at the end of each triangle-triangle test we either have one LMD pair, or the current triangle pair was rejected in one of the previous steps.

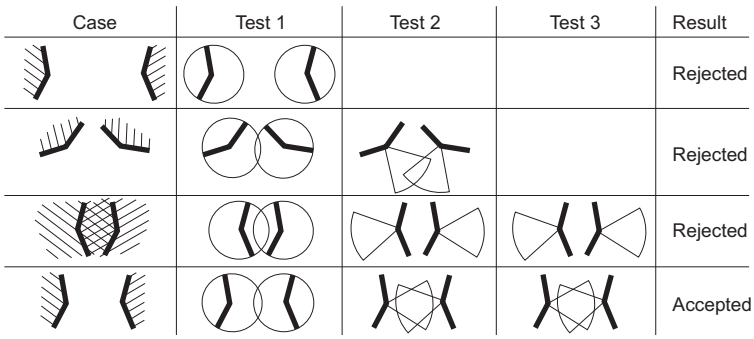

Figure 4: *Tests performed on two SNCH nodes containing objects with simple geometries. Four different cases are presented, with one passing all the tests and each one of the rest rejected by a different test.*

As the result of the complete global search computation, we get a set of the triangle pairs forming the set of active LMD pairs. However, because the global search takes usually a long time, a secondary process manages the set of LMD pairs and, using local information it updates LMD set as the user moves. This process is called *local tracking*.

3.2.2. Local Tracking

The local tracking algorithm is relatively simple. It takes the set of active LMD pairs and updates each LMD pair separately to assure that it still represents the minimal distance in the local neighbourhood. The algorithm proceeds as follows:

For each LMD pair:

L1. For contact points A and B of LMD pair, determine the neighbouring triangles according to the closest features (Fig 5). The results are triangle sets T_A and T_B.
L2. Calculate distances of all combinations of triangles in T_A set vs. T_B set.
L3. If the shortest distance pair matches original A and B points, quit.
L4. Otherwise change LMD pair to the new triangles and/or contact points a continue with step **L1**.

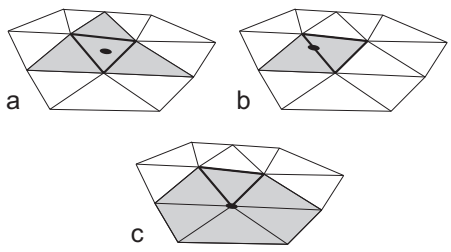

Figure 5: *The set of neighbourhood triangles that are examined, depends on the location of contact point: (a) the contact point inside the triangle, (b) on the edge of the triangle, (c) on the vertex of the triangle.*

After each LMD pair is updated, the set of LMDs corresponds more closely to the current positions of both objects. The local update is performed for each haptic frame until the new results from the global search are available.

4. An Improved Algorithm

In the following parts we will depict the differences between our algorithm and the original one, described in the previous section. In our algorithm, we also use SNCH structure and the global and local update loops. Using this framework we implemented the following changes.

4.1. Global Search Differences

The top-down traversal for global search is organized similarly. The non-leaf node tests (**G1**) and (**G2**) are the same and the **G3** test is optimized. Instead of using the original conditions, the nodes pass the third test only when satisfying the following tests:

Let $\sigma_{\Phi_A\Phi_B}$ be an angle characterizing the mutual position and size of two bounding spheres A,B with radii $\rho_{\Phi_A},\rho_{\Phi_B}$ and centers S_{Φ_A},S_{Φ_B}.

$$\sigma_{\Phi_A\Phi_B} = \arcsin\left(\frac{\rho_{\Phi_A}+\rho_{\Phi_B}}{\|S_{\Phi_B}-S_{\Phi_A}\|}\right)$$

Test **G3** is replaced by the following conditions:

G3.a.

$$\arccos\left(\frac{S_{\Phi_B}-S_{\Phi_A}}{\|S_{\Phi_B}-S_{\Phi_A}\|}\cdot\vec{C}_{\Phi_A}\right) > \sigma_{\Phi_A\Phi_B}+\phi_{\Phi_A}$$

G3.b.

$$\arccos\left(\frac{S_{\Phi_A}-S_{\Phi_B}}{\|S_{\Phi_A}-S_{\Phi_B}\|}\cdot\vec{C}_{\Phi_B}\right) > \sigma_{\Phi_A\Phi_B}+\phi_{\Phi_B}$$

The leaf tests **G4** and **G5** are also different. Instead of approximating the triangle distance by choosing the smallest of 7×7 distances, we decided to calculate the exact triangle-triangle distance. It is necessary to cope with situations when

sizes of triangles differ considerably, i.e. when a small triangle gets close to a large triangle. However, as this precise calculation is more complex than the original approximation, we added a pre-test. Using the vertex normals for projection we check whether the triangles are in such position that they may "see" each other.

This test is basically a visibility test: For each triangle A we project its vertices on the plane of the other triangle B. The projected vertices form a new triangle A_1. If the A_1 and B do not intersect, the original triangle A does not "see" the triangle B. Therefore they do not form a local minimum and it will be found somewhere else in the neighbourhood. The same test is then performed in reverse with B projected on the plane of A. The test is illustrated on Figure 6.

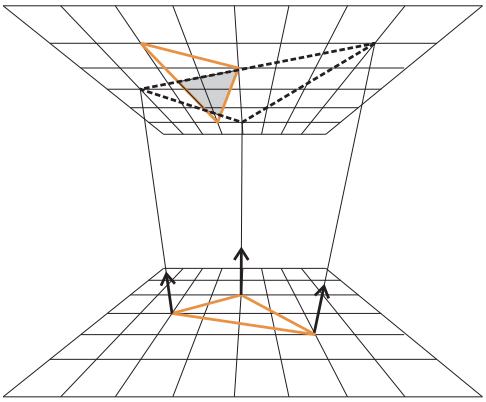

Figure 6: *A "visibility" test which detects whether two triangles are properly positioned to form a LMD pair. If the triangles in the plane do not overlap (grey area), the triangle pair is culled.*

Due to the pretesting, the resulting shortest distance vector does not need additional verification (test **G5** in original alg.) after the triangle-triangle distance calculation.

4.2. Local Tracking Differences

The main idea of the local tracking is the same, but the actual solution contains several differences:

1. Same as for the global search, we calculate precise triangle-triangle distance, instead of performing an approximation.
2. To speed up the tracking we use the cache of triangle distances already calculated in the same local update loop.
3. The cache is used not only to avoid unnecessary calculation, but also to find neighbouring duplicate LMDs and eliminate them from further processing.

Using the cache brings a significant performance increase which we will discuss in the following section.

4.2.1. Results Cache

Local loop is used to update the LMD set reflecting movements of rendered objects. The cache holds intermediate results of the triangle-triangle distance calculations: For each triangle pair in cache, it stores the distance, the nearest points and the information about the closest features (vertex-vertex, edge-vertex, etc.). Additionally, it holds the identification of the LMD pair which initiated the calculation of the respective results.

The content of the cache is erased at the beginning of each local update loop. For each candidate pair in LMD set, whenever a new triangle-triangle distance calculation is required, the cache is used to reduce the number of expensive calculations. It is searched at first for result calculated formerly when testing the neighbours of another pair. If no result is found, the distance is calculated and triangle pairs in local neighbourhood of both triangles are also tested to find a local minimum. These pairs with results are also stored in cache. The new local distance minimum which may be located on a different pair, is used to update the LMD set.

The second reason for using cache is even more important. Because the algorithm now possesses the information about previous computations, it can easily detect whether a tested LMD pair is a neighbour of an already calculated one. This is detected when there is a request to test triangle pair that was already tested for a different LMD pair. If such case is encountered, the algorithm still finishes calculations of neighbouring triangle pairs distances and chooses the shortest one. If the shortest one is the same as the one already found in cache, the algorithm discards the current LMD pair in LMD set completely. The idea is that if a pair with this distance is in cache, it was already compared to its neighbourhood and this situation is reflected by previously processed LMD pairs. In most cases, this optimization reduces the number of tracked LMD pairs by an order of magnitude in high-poly models.

4.3. Local Tracking Inside the Global Update Thread

The tracking part consists of two overlapping phases: global update prepares a new LMD set from scratch and this set is many times updated when tracking the collision pairs locally. However, for large number of LMD pairs the local update calculations and management may be expensive. We noticed, that the first local update called immediately after the global search finished preparing a new LMD set, tends to remove large number of LMD pairs, which is a computationally demanding process. Therefore we appended local update step at the end of the global search. Its main goal is to reduce LMD set significantly before entering local update loop. We simply call the function for local tracking in the "global update" thread immediately after finishing the global calculations and we call this function twice. As we already mentioned, the time required to finish the first local

update may be long and may even equal the time required for the global update itself. During this time the reduced LMD pairs may be already too distant from the real user's position. Passing the reduced but late results directly to the local tracking thread would also slow down its calculations. This is the reason why we execute the local tracking function in the global thread for the second time. The second call does not aim to reduce LMD set, but corrects the LMD pairs to reflect a new position better.

4.4. Intersection Calculation

A side effect of more precise approach in finding LMD pairs is, that it not only calculates distances, but can also produce precise intersection polylines. Though we do not use it currently for haptic rendering, we can exploit it when simulating interactions between virtual tools and objects with complex shapes.

4.5. Force Calculation

In contrast to the original approach, we allow LMD pairs to represent stronger and smaller virtual strings generating force with much sharper gain. Therefore users can move the object that they are holding, closer to the obstacle and can also perceive smaller features on the surfaces. This is possible due to more precise calculations of the LMDs.

Also, when converting distances to force we do not use a simple linear conversion, but an exponential one. Maximum force is achieved at some distance from the surface to prevent the user from getting too close.

The force vectors derived from LMD pairs are then summed up and the size of the final vector is clipped to the largest force. This simple countermeasure effectively solves the situations when there are many LMDs in the same direction.

5. Implementation

For testing we used a computer equipped with dual AMD Opteron 275 (i.e. total of four cores), 1 GB RAM and NVIDIA Quadro4 750 XGL graphic card. For haptic interaction we deployed the ReachIn Display station with a Sensable^{TM}PHANTOM®device capable of detecting 6 DOF user movement and 3 DOF force on output. Due to this limitation, we were not able to simulate torque, but only a translational forces.

The haptic visualization is implemented using OpenHaptics Toolkit and is included in a component system called VRECKO. as one of its core parts. Together with a space subdivision component it is able to track user's movement through a world composed of arbitrary number of objects.

In our system, we use four threads to perform the critical tasks in parallel when possible. The threads are assigned one of the following operations:

1. Scene management and graphics rendering
2. Phantom loop
3. Global LMD searching
4. Local LMD tracking

The first thread performs the routine tasks that any simulation system has to do. Because the collision calculation are offloaded to other threads/processors, the system feels very responsive.

The second thread runs a loop to update main object position according to the user movement and also to calculate the final force. This loop is usually running at frequency of 1000 Hz.

The third thread runs a global loop which constantly tries to find potential future collision points - the LMD pairs. In every loop, the points are calculated "from the scratch", discarding any previous information. This full calculation is usually a slow operation, therefore the thread can sometimes make only a several loops per second. Its results are transferred to the fourth thread.

The fourth thread performs the local tracking, i.e. it maintains the current set of LMD pairs and updates their positions according to the object movement. The results of this thread are passed to the Phantom loop (the second thread).

Using a computer with two dual-core processors effectively means that each of four main threads is running on its own processor core. The threads are forced to run on a specific processor core, bypassing the main scheduler. Therefore the global search and local tracking threads run as fast as possible without any interruptions from the graphic thread or from each other.

6. Practical Experiments

To fully capture the difference between the original algorithm and our improved version, we conducted several different tests in which various aspects of the methods were considered.

We have tested the methods on various models. For this series of tests we have chosen a scene with medium complexity: Two bunny models with one directly linked to the user's movement and the other statically positioned in the scene. Each bunny has over 10,000 triangles (a screenshot from the scene is presented on Fig 1).

6.1. Benchmarks

Firstly, to measure computational demands we set up a scene and disabled the user input and the force output. One of the bunnies was animated along a given path and timings were performed for both methods. This test was repeated with a different force settings (because of this difference a different path has to be used). The respective graphs are shown on

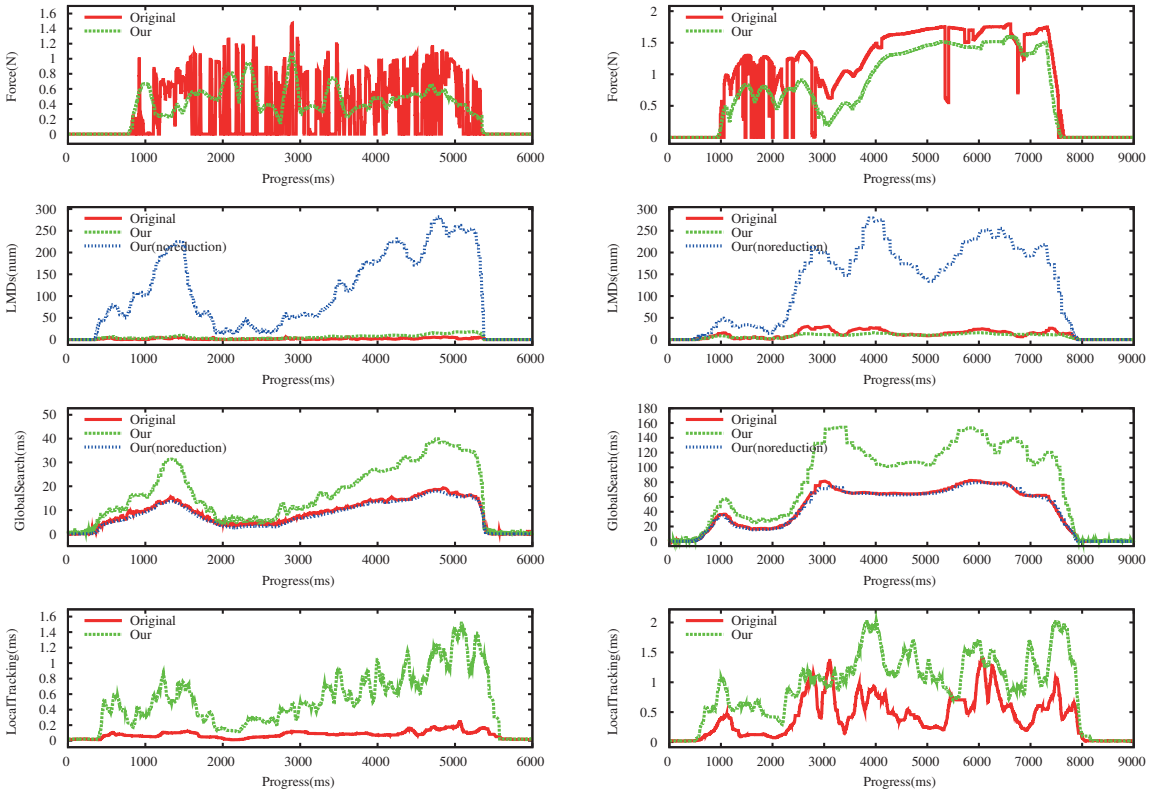

Figure 7: *Benchmarks simulating hard and soft contact on left and right respectively. For both types of contact a different path was used. For detailed discussion see section 6.1.*

Fig. 7 with "hard" force settings generating harder contacts on the left side of the figure and "soft" force on the right side.

From the graph several effects can be observed. Using the hard force our method shows good force stability, but the original method jumps quickly from zero to strong force and back. This was expected and the reason is that the working set of LMDs is relatively small (under 10) and due to the approximations the method quickly discover and lose LMDs almost randomly. This leads to dramatic changes in the force. This behaviour improves when using soft force and the objects are kept at a greater distance. The LMDs are still disappearing and returning back, but their total number is greater and due to the greater distance between objects the force contribution of one LMD is not so significant as in the previous case.

Also from the force graph we can see that the resulting forces when using the soft settings are very similar with both methods. The original method however generates slightly higher force. This is due to the fact that we are using exponential force gain instead of linear. With a few parameter changes both methods can generate a force that resembles the other one more closely.

The next graphs show the number of LMD pairs. Both methods maintain similar number of LMD pairs, though the original method fluctuates more wildly. There is also depicted the number of LMD pairs that are found by our method before the reduction. Usually this number is very high and tracking all such LMD pairs would be nearly impossible in real-time. Fortunately, the reduction process is very efficient and is able to reduce the total number of LMD pairs up to twenty times in these situations. Of course, this depends on the mesh complexity - with more complex meshes the reduction factor is greater.

The global search time is worse with our method. The "base" time is almost the same, but the difference lies in the reduction process after the end of global search, which can take the same time as the global search itself. The result is that the total time of our global searching + first reduction is approximately two times longer than the original one.

Unfortunately, the local update time is also worse with our method and it appears to be much worse with the hard force simulation. On the other hand, we have to keep in mind that the original method skips valid LMD pairs in the global search thus having less calculations to perform in the local

update. When relaxing requirements to soft force interaction, the situation is more optimistic. The original method is still in the lead, but the difference is not as dramatic as shown on the previous graph.

Aside from the presented graphs we have also measured the load of individual CPU cores. The measurements showed that Phantom loop is consuming much less processing power than other threads and it can run together with local LMD tracking on one core.

6.2. Users Experiences

While the method described in this paper is computationally more expensive and thus slower than the original one, it can give the feeling that is more faithful. The original method is fast, but due to approximations when testing the triangle pairs for proximity, the resulting force is less realistic and provides only "soft" guidance for the user. Using more precise calculations to estimate the proximity of objects gives a "sharper" haptic sensation and the user can feel surface features much better.

We have used the same tests as in the benchmark section, but with a free movement of a user, turning the force feedback on. The feeling with the hard force is significantly different with both methods. Compared to our method that is able to track distances between objects more accurately, the original one causes jumps between vertices pairs and the user experience with sudden changes of force feedback is erratic.

On the other hand, when using soft force, the original method may perform better. Because the demands on precision are much lower in this case, the original method runs faster on smooth meshes with higher level of detail without sharp edges. Our method can simulate the same effect even for the meshes with triangles of differing sizes, but it requires more computational power.

7. Conclusion and Future Work

We have presented a method for haptic rendering of collisions between complex models. Though similar method already exists, it was targeted only for soft contacts acting only as a guide for proper placement. Our improvements of this method made it useful also for evaluating hard contacts. Hence, the user is able to feel better the surface of the colliding objects. We achieved it using the greater precision of internal calculations. The resulting algorithm is slightly slower; however, the loss of performance is not dramatic due to the various optimization techniques used, most notably the cache and the associated process of contact pairs reduction. Considering the global search, it is challenging to implement part or all of it using the CUDA architecture to accelerate the calculations using the graphic card processor. The local tracking would not probably benefit from CUDA implementation, because of introduced latency. However, it may be advantageous to split this search into several threads and also to introduce estimation of user's future movements. On the lowest level, we want to improve the force calculation algorithm. While the current version works reasonably well, it may have problems in complicated situations with many LMD pairs pointing in the same direction. Furthermore, it would be useful to add a support for dynamic meshes. Currently, it would require to rebuild the whole object hierarchy, which is a time consuming process, but may be feasible for local changes.

8. Acknowledgements

This work was supported by the Ministry of Education, Youth and Sports of the Czech Republic under the research program LC06008 (Center for Computer Graphics).

References

[Bar96] BARNER F. .: Haptic scientific visualization.

[BJ07] BARBIČ J., JAMES D. L.: Time-critical distributed contact for 6-dof haptic rendering of adaptively sampled reduced deformable models. In *2007 ACM SIGGRAPH / Eurographics Symposium on Computer Animation* (Aug. 2007).

[BJ08] BARBIČ J., JAMES D. L.: Six-dof haptic rendering of contact between geometrically complex reduced deformable models. *IEEE Transactions on Haptics 1*, 1 (2008), 39–52.

[JC01] JOHNSON D. E., COHEN E.: Spatialized normal cone hierarchies. In *ACM SIGGRAPH Symposium on Interactive 3D Graphics (I3D)* (2001), pp. 129–134.

[JW03] JOHNSON D. E., WILLEMSEN P.: Six degree-of-freedom haptic rendering of complex polygonal models. In *Haptic Interfaces for Virtual Environment and Teleoperator Systems* (2003), pp. 229–235.

[JW04] JOHNSON D. E., WILLEMSEN P.: Accelerated haptic rendering of polygonal models through local descent. In *Haptic Interfaces for Virtual Environment and Teleoperator Systems, 2004. HAPTICS '04. Proceedings. 12th International Symposium on* (2004), pp. 18–23.

[JWC05] JOHNSON D. E., WILLEMSEN P., COHEN E.: 6-dof haptic rendering using spatialized normal cone search. In *IEEE Transactions on Visualization and Computer Graphics* (2005), pp. 661–670.

[KOLM02] KIM Y., OTADUY M., LIN M., MANOCHA D.: Sixdegree -of-freedom haptic display using localized contact computations, 2002.

[OL03] OTADUY M. A., LIN M. C.: Clods: Dual hierarchies for multiresolution collision detection. In *Eurographics Symposium on Geometry Processing* (2003), pp. 94–101.