# Perception-influenced Animation

Daniel Schiffner, Detlef Krömker
Professur für Graphische Datenverarbeitung
Goethe Universität Frankfurt
Robert-Mayer-Str. 10
60054, Frankfurt (Main)
{kroemker,dschiffner}@gdv.cs.uni-frankfurt.de

**Abstract:** The heterogeneity of future living environments will increase the necessity to create applications that can run on any device. In the context of graphics applications, some kind of simplification must be included to enable rendering on devices with less computation power. Using perception to guide such a simplification is a common approach. However, existing methods generate levels of detail in advance, and only a selection is performed during run-time. In simulations, this not sufficient because an object will change over time.

We present a framework that adapts a simulation using perceptual measures. We use a visual salience model to extract regions where detail can be modified. This information is calculated during run-time, and by using a dynamic data structure, the representation is adapted without a definition of levels of detail in advance. We included the system in a physics library and so created an interactive and continuous simulation level of detail.

## 1 Introduction

Physical behavior of objects in a 3d scene create the impression of a real-world scenario. Simulations of fabrics and liquids, for example, require expensive calculations, and if accuracy is not crucial, a reduction in the number of simulated objects leads to a decrease in computation time. In real-time applications, such as games, a reduction may be applied as long as the plausibility is retained, e.g. a human would rate the visible outcome as valid. In this work, we propose to use perceptual information when altering detail, and we present a framework to modify the detail of a simulation during run-time. This adaptation is not bound to a predefined set of simulation levels of detail (SLODs). Thus, it can adapt to any hardware without manual adjustments. With this system, it is theoretically possible to transfer a running simulation from a high-performance system to another, maybe lower-powered device. A user is no longer bound to a specific hardware at a distinct location.

By simulation, we thereby mean changes to the surface representation based on physical laws. These changes, for example, can be introduced due to gravitation or compression of an object. Usually, objects are decomposed into multiple parts to increase detail, i.e. accuracy, of the simulation. Our framework will control this decomposition, and to complete this task, perceptual measures are used to steer the applied modifications, e.g. a reduction or increase in detail.

To account for perceptual information, a visual salience model is used. An object is considered salient if it figuratively "pops out" of its surround. We utilize a 3d visual salience model and include animation-specific features, e.g. motion. The Bidirectional Saliency Weight Distribution Function (BSWDF) [SK11] allows us to extract regions of interest within a 3d scenario, and thus we can account for areas that are important for a human spectator. Due to the inclusion of animation-specific features, regions that are modified by the simulation are taken into account as well.

We derive a Model-View-Controller-system that extracts, alters, and displays simulated objects. Our prototype is based on a soft-body simulation. A soft-body, which covers materials like cloth or fabrics, is well suited for dynamic adaptation as individual parts of a soft-body-simulation can be removed to reduce the accuracy. Our results show that real-time updates can be achieved using nowadays hardware.

After giving this introduction, we continue to look at related work. In section 3, we present our framework. Afterwards, the saliency features are derived, and the according BSWDF is defined. In section 5, we give some notes regarding the complete system and present performance measures as well as achieved results in section 6. We close with a conclusion and present intended future work.

## 2 Related Work

Soft-body objects have a high computational demand, and these are neither rigid, fluid, nor gaseous. To reduce simulation complexity of fluids and gaseous materials, point-based animations have been studied by several researchers. Mueller et al. [MKN+04] presented a separated data layout to reduce the size of the simulation data. Adams et al. proposed the "adaptively sampled particle fluids" approach [APKG07]. Single nodes are collapsed or expanded to reduce the size of the simulation. Similar to Mueller et al., a separated data layout is chosen.

The accuracy of a simulation can be modified during run-time by using multi-resolution representations. Beaudoin and Keyser [BK04] replace simulations of plants with approximations. These approximations are generated during a preprocess, and individual parts of a plant are exchanged using a recursive algorithm. Visual artifacts are avoided with smooth transitions. This geometric approach is both a LOD and SLOD as the detail in the representation and the simulation is reduced. It, however, requires to traverse the representation each time it is accessed, and the recursive algorithm does not account for perceptual information.

The perception of a human spectator can be modeled with visual salience as done by Itti et al. [IKN98]. Their method represents the processes of early vision, which cannot be influenced by tasks or other cognitive-related factors. For rendering, saliency information can be used to preserve detail in important regions [SK10, FSG09, LVJ05].

In [SK10], we presented a dynamic data structure, which selects a representation for rendering using a priority value. Carmona and Froehlich [CF11] proposed a similar approach simultaneously, and they presented a theoretical optimal algorithm for priority selection.

This priority, for example, can be saliency information, such as curvature of the surface. For general computation of saliency within a 3d scenario, the BSWDF has been defined [SK11]. This allows to model visual salience of an object without explicit projection into 2d space. In combination with the *TreeCut*, a perception-based LOD is established.

Some user studies regarding simulation and perception have been performed by several researchers [YRPF09, GDO08, O'S05]. The results of the different experiments show that the precision of physics simulations can be reduced without loosing plausibility.

# 3 Approach

Our approach is a combination of the *TreeCut* data structure – presented in [SK10] – and a soft-body simulation provided by the Bullet physics library [Bul]. We have chosen the library to show, on the one hand, the universal applicability of our approach, and on the other hand, avoid the need to verify a stand-alone physics calculation.

To establish this combination, both the *TreeCut* and the physics library need to be extended, so that a dynamic exchange of information between both is possible. Furthermore, to account for saliency information, a specialized BSWDF will be defined, which uses animation-based features, such as motion. While the BSWDF is not limited to these animation features, we will focus only on these within this work.

Our framework is depicted in figure 1. The complete system is highly dynamic as the change introduced by the simulation invokes new changes in the *TreeCut* representation. We use the simulated data to derive both the visual output (View) and the influence of visual salience computation (Feedback Stage). We control the Feedback Stage of the system with a threshold value for saliency values, and so limit the adaptation of simulation detail.

## 3.1 Physics Simulation

The Bullet physics library [Bul] provides various tools and data structures to compute a physics-based simulation. This includes detection and resolving of collisions as well as physics objects, e.g. rigid- and soft-bodies, as well as their computations

In our prototype, the Bullet library in version 2.78 is used. It provides a soft-body simulation class, which is supplementary to the normal library. We leverage this fact to plug-in our own *TreeCut*-SoftBody (TC-SB) that bases on the Bullet's `SoftBody`-class. It utilizes a Mass Spring System (MSS) to simulate the physical behavior. Only two data structures – the simulation nodes and links – are required for the computation of forces.

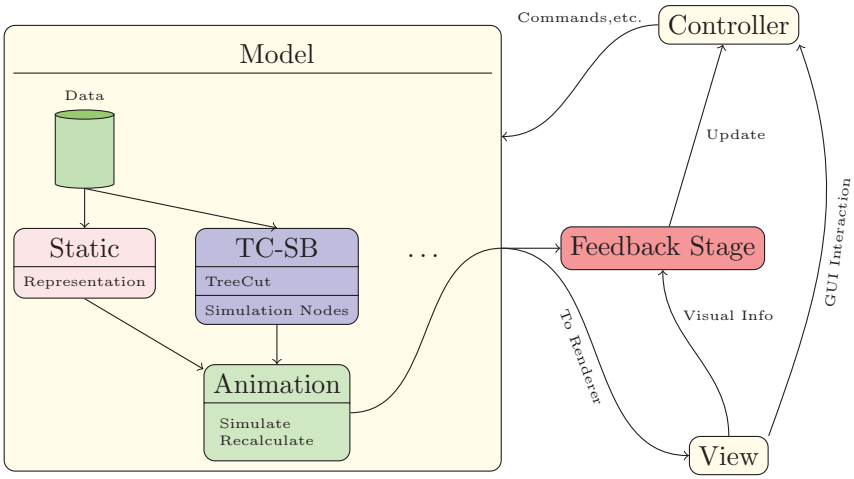
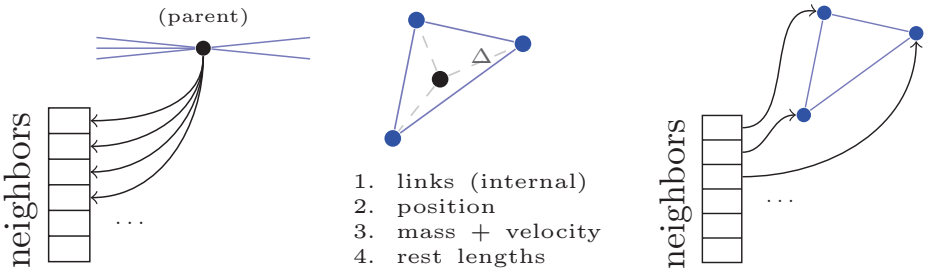



Figure 1: The proposed animation framework that is based on a Model-View-Controller-design. The Model contains the dynamic representation using a *TreeCut* and a soft-body simulation (TC-SB). Other representations can be added as well, which will be included in the animation. The Controller issues the *TreeCut*-operations based on the results from the Feedback Stage. The View is derived from the current representation of the Model and can introduce, via the user, new changes into the system.

### 3.2 *TreeCut*

The *TreeCut* utilizes a multi-resolution representation of an object. With the help of two core operations, `refine` and `coarse`, the detail of a local node is altered. In case of a `refine`-operation, a single node is replaced with a more-detailed representation. Assuming a tree, a node is replaced with its children. The `coarse`-operation is the inverse, i.e. the children are replaced with their common parent. We use a priority-selection of surface elements, so called surfels, to invoke a *TreeCut*-operation for the assigned node.

The *TreeCut* has been defined for geometrical LOD-methods, and thus both operations need to be extended to correctly account for SLOD-operations. Physics calculations need access to the surfel structure maintained by the *TreeCut*. We therefore use an index-based mapping between the surfels and the simulation nodes. The MSS-simulation uses links, and the inner structure of a mesh needs to be reestablished after a SLOD-method has been applied. Because of the locality of the *TreeCut*-operations, we include an incidence list, which avoids searching for connected links. An interpolation between the old and the new state of the MSS is done to reduce artifacts.




(a) Extract the neighbors that are adjacent to the parent node via the links. Also, mark these links for interpolation.

(b) Set the simulation parameters. Internal links (between children) are created. Masses, velocities, and rest lengths are calculated. $\Delta$ is the added displacement.

(c) Connect external nodes using the nearest node in the neighbor list. Mark these links for interpolation.

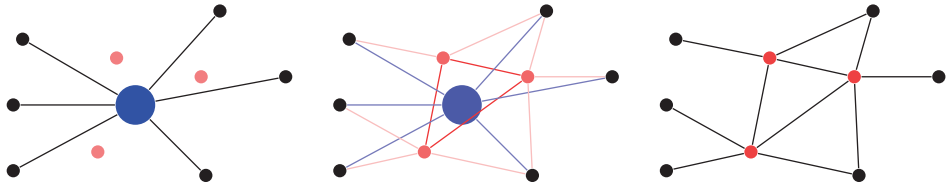
Figure 2: The `refine`-operation and the required steps to assure correct generation of simulation nodes. In the first step, the incident links are extracted while the physical properties are propagated to the child nodes in the second. In the third, the external links are distributed among the children and interpolation is enabled.

### 3.3 Beginning TreeCut-Operations

In the following, we assume a `refine`-operation to be processed – only small adaptations are required to perform a `coarse`-operation. To replace a simulation node with its successors, the following steps are performed:

1. Store (old) incident links
2. Propagate physical properties
3. Create (new) incident links
4. Mark links for interpolation

When `refine`-ing a node, the incident links are marked for interpolation and adjacent nodes are extracted. The physical properties of the parent node are acquired and propagated. For the position information of the children, a displacement relative to their parent is calculated. The child nodes are added to the set of simulation nodes, but are excluded from collision calculations, for now. After propagation, the new nodes are connected with their neighbors using local nearest neighbor search among the adjacent nodes. The created links are added to both incident link lists of the affected nodes and are marked for interpolation. In figure 2, the applied steps for propagation are visualized while figure 3 shows the interpolation.



(a) The current links to the node (blue) will be marked for delete and are interpolated due to the `refine`. The new nodes (red) will be connected with the neighboring nodes.

(b) The influence of the old links (blue) is reduced during the interpolation while the new links (red) gain importance. Inner links are excluded from blending.

(c) The simulation nodes after the applied `refine`-operation. The old nodes and links have been removed and the new links are no longer interpolated.

Figure 3: Application of the blending during the *TreeCut*-operations. It is performed to avoid the generation of artifacts. From left to right, a `refine`-operation is performed. From right to left, it is equivalent to the operations performed by a `coarse`-operation. In the captions, the `refine`-operation is explained.

### 3.4 Ending TreeCut-Operation

After completion of an interpolation, old nodes and links are removed from the simulation. We apply the following steps to complete a *TreeCut*-operation:

1. Finalize physical properties of the new nodes (e.g. insert collision shapes)
2. Delete old links (in sequential order)
3. Delete old nodes

First, the physical properties of the new nodes are finalized. Therefore, the collision shapes are inserted into the simulation. The incident links are no longer interpolated.

The incident link list of the old nodes is used to extract the affected links that will be deleted. To safely remove links, their internal order may not be altered. Otherwise, the calculated position will differ because links are evaluated sequentially. Therefore, a linear traversal is required for deletion, increasing the theoretical time complexity of the end-algorithms from $O(1)$ to $O(L)$ where $L$ is the number of links. However, these links will be deleted in a lazy manner, and thus the overall time for computation is not influenced. After marking the links for removal, the nodes are removed as well.

## 4 Animation Features and Saliency

Animation features are accounted for by extracting local and global motion information that is generated during the simulation. These features will be stored in the surfel structure.

This allows combination with visual features as the BSWDF can be calculated during rendering.

The result of the BSWDF is a priority value that reflects the relative importance of a surfel. The *TreeCut*-evaluators use this priority to perform a reduction in simulation detail.

#### Definition of the BSWDF

Local motion is the relative motion of a node with respect to its previous position, i.e. its velocity. If a single node is moved differently than others, it is considered salient. This definition matches the processing of the human visual system (HVS). A `refine`-operation applied to that node will increase detail. In regions where no explicit nodes are found, the `coarse`-operation can safely be applied because it is unimportant for the HVS.

Global motion increases the saliency values as movable objects catch one's attention. However, this increase simultaneously limits the ability to focus an object [YPG01]. Thus, the maximal saliency value is clamped to an upper bound influenced by object's velocity.

We define a BSWDF that operates on both motion features. Global and local features are separated because the global features influence the local ones. Because of the restriction to two motion-related features, we state that other features need to be included when performing visual tests. In this special case, however, we give the following definition:

$$\text{BSWDF}(\omega_C, \omega_L, d_C, \vec{x}) = \text{Illu}(\omega_L, \vec{x}) \circ \text{Animation Features}(\omega_C, d_C, \vec{x}) \tag{1}$$

with $\omega_L$ being the light, $\omega_C$ the camera in spherical coordinates and $d_C$ the distance to the camera. The $\circ$-operator applies the illumination model. In case of a Lambertian illumination, this is a multiplication. We define the Animation Features as

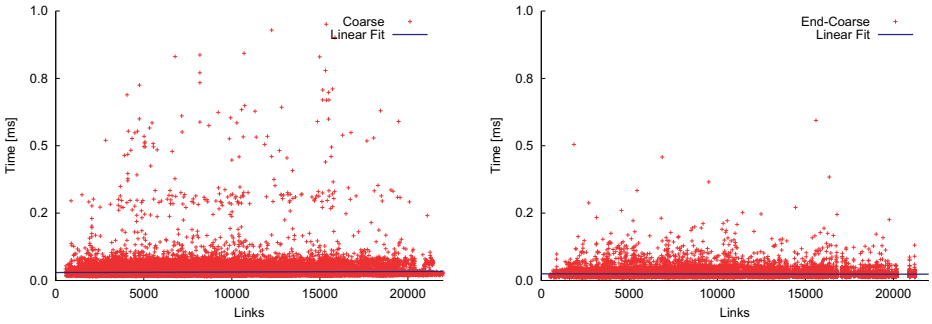
$$\text{Animation Features}(\omega_C, d_C, \vec{x}) = \text{Global Motion}(\omega_C, d_C) \otimes \text{Local Motion}(\omega_C, d_C, \vec{x}) \tag{2}$$

The position $\vec{x}$ is the current surfel's position assigned to a simulation node. If no illumination is used, the BSWDF simply evaluates to the result of the Animation Features. The $\otimes$-operator expresses the before mentioned limitation of the Local Motion features due to the Global Motion.

## 5 Complete System

The animation framework, as shown in figure 1, contains the physical calculations as part of the Model. This allows to include the extensions into an existing MVC-design to provide both physical simulations and perceptual evaluation.

The Feedback Stage receives input from the Model and the View to calculate the BSWDF.




(a) The `coarse`-operation

(b) The end-`coarse`-operation

Figure 4: The time complexity of the `coarse` and end-`coarse`-operations applied by the *TreeCut*-SoftBody. A linear fit is shown that uses all generated samples. For the `refine`, similar timings are achieved.

The *TreeCut*-evaluation is performed in parallel, and the according changes are sent to the Controller. This executes the `refine`- and `coarse`-operations.

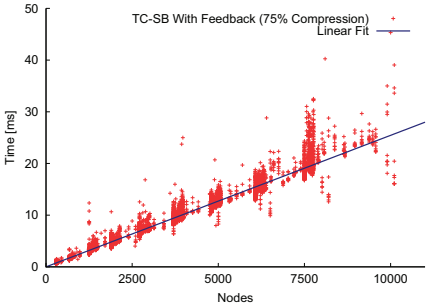
To allow adaptation of an object, some kind of restriction has to be imposed. In our prototype, we use a maximal node count, but also a maximal computation time or hardware capabilities can be utilized. For example, a maximal calculation time allows to adapt the simulation to achieve interactive rendering.

The complete system is highly dynamic and provides self-optimization capabilities. We include a threshold during *TreeCut*-evaluation to avoid repetitive change of a single node. For example, a `coarse`-operations could remove a node, which will be inserted in the next iteration again. With the threshold, the gain of inserting a node must be higher than the penalty of removing another node including the threshold.

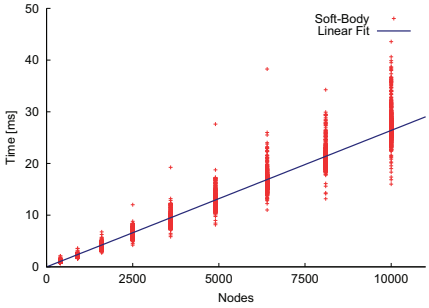
## 6 Performance and Results

We have implemented a prototype of the proposed system in `C++`. A cloth object is generated using a regular grid and the required LOD-hierarchy is generated in a preprocess. The corners are fixed to create a swinging net. In the tests, only gravitation is simulated. All results were taken on a system with an Intel i5 670 at 3.47 GHz, 8.0 GB RAM and a nVidia GeForce 260 GTX graphics card.

In a first test, we measured the performance of the dynamic SLOD-methods. In table 1, the manipulation of a single node is presented averaged over multiple applications of each *TreeCut*-operation. Each entry in the table contains the fraction of the complete processing time along with its measured times.




(a) *TreeCut*-SoftBody-simulation with Feedback enabled. The simulation is compressed to 75% of the initial size.




(b) `Soft-Body`-simulation. No compression can be applied, and thus the node count is constant.

Figure 5: The performance of the simulation in dependency of the node count used. The graphs plot the enhanced *TreeCut*-SoftBody and the plain `SoftBody`, i.e. where no compression is applied. The results are taken over a complete test set.

The *TreeCut*-operations have a high overall performance. Due to the neighbor information, both *TreeCut*-operations remain constant in time (see figure 4).

In a second test, the simulation time in dependency of the number of simulation nodes and links is evaluated. A test set is defined, which consists of multiple nets with varying node counts and compression rates. During each test, no collisions are performed.

The averaged timing result (see table 2) show that the *TreeCut*-SoftBody decreases the node count and increases performance. In the results, a reduction in processing time to approximately 47.76% is achieved (compression to 30%) despite the additional 25 interpolations performed each step. Figure 5 shows the linear dependency in node count of the *TreeCut*-SoftBody.

In figures 6a to 6j, some images generated with the *TreeCut*-simulation are shown. For comparison, the results achieved with the Bullet's `SoftBody` are included. The default `SoftBody` does not have the ability to change the SLOD. If a lower detailed version is required, a new `SoftBody` has to be created accordingly.

| Operation | N | L | Node-Ops [ms] | Link-Ops [ms] | Complete [ms] |
|---|---|---|---|---|---|
| `refine` | 4888 | 10419 | 0.0164 (91.14%) | 0.0016 (8.94%) | 0.0180 |
| end-`refine` | 4889 | 10794 | 0.0017 (94.39%) | 0.0001 (5.53%) | 0.0018 |
| `coarse` | 4966 | 10615 | 0.0319 (97.61%) | 0.0007 (2.39%) | 0.0327 |
| end-`coarse` | 4938 | 10917 | 0.0025 (99.20%) | 0.000 (0.79%) | 0.0025 |

Table 1: Different timing results when applying the *TreeCut*-operations to the simulation. N denotes the average number of nodes present while L is the number of links. The Node-Ops are all operations that affect a node of the simulation while Link-Ops modify its links.

| Method | Compression | N | L | I | Total [ms] |
|---|---|---|---|---|---|
| Bullet SoftBody | - | 4266 | 8413 | 0 | 11.09 |
| TC-SB | NoCompression | 4266 | 8647 | 0 | 10.08 |
| | 75% | 3324 | 6931 | 30.5 | 8.40 |
| | 30% | 1911 | 4220 | 25.1 | 4.80 |

Table 2: Timing results with and without the feedback-loop using varying node counts. Compression is the fraction the initial size is reduced to. N denotes the node count, L the number of links present and I the number of interpolations. All results are averaged over a complete test set.

## 7 Conclusion and Future Work

The presented system simulates a soft-body object, and the number of simulation nodes can be reduced during run-time without a definition of discrete levels in advance. As opposed to existing methods, we do not require a separated data layout. A SLOD-reduction is applied based on visual salience with animation-specific features. The representation can adapt to any hardware.

Unlike full point-based approaches, the removal of simulation nodes may not be performed without preconditions. We store incident links to circumvent restrictions and enable dynamic adaptation using the *TreeCut*.

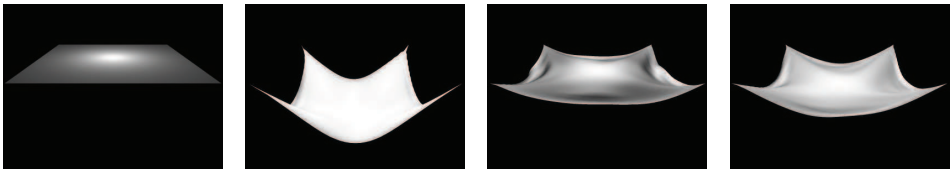
We defined a special BSWDF that operates on animation features, such as local motion. The results are used to control the dynamic adaptation via the *TreeCut*-evaluation. Because of the BSWDF, a human-oriented adaptation is performed.

Currently, only one *TreeCut* is applied for both rendering and simulation. We plan to apply a second *TreeCut*, which will maintain the simulation nodes. It will then be necessary to propagate the results of the simulation nodes to the surface representation.

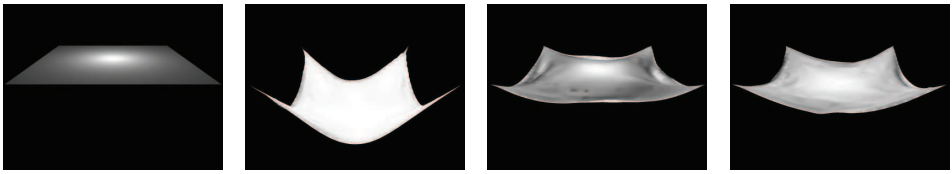
When applying a *TreeCut*-operation, the placement of new nodes is performed using only geometrical measures. This needs to be extended with animation and physical measures to increase stability and feasibility. Also, a smooth surface, e.g. a moving least squares surface, could provide more exact positions.

An implementation using the graphics card could increase performance. With the capabilities of the newest shader model 5, a soft-body can be implemented without restrictions. However, it remains to be seen what changes to the proposed system are required.

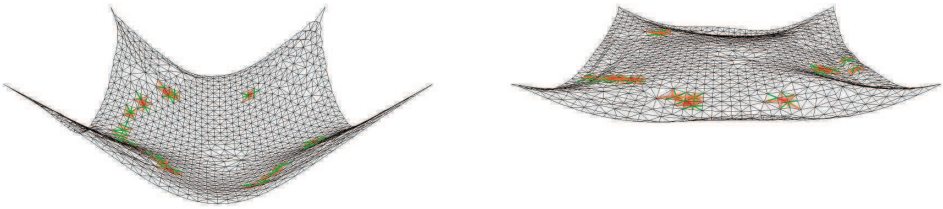
Our system can account for perception when evaluating a simulation object. User tests have to be performed to validate the gained impression that detail is preserved and that the simulation remains plausible even if nodes are `coarse`ned or `refine`d. A maximal compression factor could be derived to identify when a simulation looses its plausibility.



(a) `SoftBody` initial configuration (b) After 45 steps (c) After 90 steps (d) After 180 steps



(e) *TreeCut* initial configuration (f) After 45 steps (g) After 90 steps (h) After 180 steps



(i) *TreeCut* links after 45 steps (j) After 90 steps

Figure 6: Comparison of the `SoftBody` and the *TreeCut*. A fixed time step is used to generate the different configurations. In the last row, the links of the simulation are visualized. The highlighted links are being replaced by the *TreeCut*-operations. In case of the *TreeCut*-SoftBody, only an approximate surface reconstruction is applied. Yet, the detail is retained in the fast moving center region.

## References


[APKG07] Bart Adams, Mark Pauly, Richard Keiser, and Leonidas J. Guibas. Adaptively sampled particle fluids. *ACM Transactions on Graphics*, 26(3):48:1–48:8, 2007.

[BK04] Jacob Beaudoin and John Keyser. Simulation levels of detail for plant motion. In R. Boulic and D. K. Pai, editors, *Computer animation 2004*, pages 297–304, Aire-la-Ville, 2004. Eurographics Association.

[Bul] `http://bulletphysics.org/`. visited on 19.12.2011.

[CF11] Rhadamés Carmona and Bernd Froehlich. Error-controlled real-time cut updates for multi-resolution volume rendering. *Computers & Graphics*, 35(4):934–944, 2011.

[FSG09] Miquel Feixas, Mateu Sbert, and Francisco Gonzalez. A unified information-theoretic framework for viewpoint selection and mesh saliency. *ACM Transactions on Applied Perception*, 6(1):1:1–23, 2009.

[GDO08] Marcos Garcia, John Dingliana, and Carol O'Sullivan. Perceptual Evaluation of Cartoon Physics: Accuracy, Attention, Appeal. In Sarah Creem-Regehr, K. Myszkowski, Bobby Bodenheimer, Betty J. Mohler, Bernhard Riecke, and Stephen N. Spencer, editors, *Proceedings APGV 2008*, pages 107–114, New York, 2008. ACM Press.

[IKN98] Laurent Itti, Christof Koch, and Ernst Niebur. A model of saliency-based visual attention for rapid scene analysis. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 20(11):1254–1259, 1998.

[LVJ05] Chang Ha Lee, Amitabh Varshney, and David W. Jacobs. Mesh saliency. In Markus Gross, editor, *Proceedings of ACM SIGGRAPH 2005*, pages 659–666, New York, 2005. ACM.

[MKN+04] Matthias Mueller, Richard Keiser, Andrew Nealen, Mark Pauly, Markus Gross, and Marc Alexa. Point based animation of elastic, plastic and melting objects. In R. Boulic and D. K. Pai, editors, *Computer animation 2004*, pages 141–151, Aire-la-Ville, 2004. Eurographics Association.

[O'S05] Carol O'Sullivan. Collisions and Attention. *ACM Transactions on Applied Perception*, 2(3):309–321, 2005.

[SK10] Daniel Schiffner and Detlef Krömker. Tree-Cut: Dynamic Saliency Based Level of Detail for Point Based Rendering. In Ralf Dörner and Detlef Krömker, editors, *Self Integrating Systems for Better Living Environments: First Workshop, Sensyble 2010*, pages 37–43. Shaker Aachen, 2010.

[SK11] Daniel Schiffner and Detlef Krömker. Three Dimensional Saliency Calculation Using Splatting. In Nenghai Yu, editor, *Sixth International Conference on Image and Graphics (ICIG), 2011*, pages 835–840, Piscataway, 2011. IEEE.

[YPG01] Hector Yee, Sumanta N. Pattanaik, and Donald P. Greenberg. Spatiotemporal sensitivity and visual attention for efficient rendering of dynamic environments. *ACM Transactions on Graphics*, 20(1):39–65, 2001.

[YRPF09] Thomas Y. Yeh, Glenn Reinman, Sanjay J. Patel, and Petros Faloutsos. Fool Me Twice: Exploring and Exploiting Error Tolerance in Physics-Based Animation. *ACM Transactions on Graphics*, 29(1):5:1–11, 2009.