A Threefold Representation for the Adaptive Simulation of Embedded Deformable Objects in Contact

Martin Seiler
ETH Zürich, Switzerland
seiler@vision.ee.ethz.ch

Jonas Spillmann
ETH Zürich, Switzerland
spillmann@vision.ee.ethz.ch

Matthias Harders
ETH Zürich, Switzerland
mharders@vision.ee.ethz.ch

ABSTRACT

We propose an approach for the interactive simulation of deformable bodies. The key ingredient is a threefold representation of the body. The deformation and dynamic evolution of the body is governed by a cubic background mesh. The mesh is hierarchically stored in an octree-structure, allowing for a fully local adaptive refinement during the simulation. To handle collisions, we employ a tetrahedral mesh, allowing for an efficient collision detection and response. We then show a physically-plausible way to transfer the contact displacements onto the simulation mesh. A high-resolution surface is embedded into the tetrahedral mesh and only employed for the visualization. We show that by employing the adaptive threefold representation, we can significantly improve the fidelity and efficiency of the simulation. Further, we underline the wide applicability of our method by showing both interactive and off-line animations.

Keywords: Physically-Based Animation, Deformable Models, Collision Handling, Finite Elements, Adaptivity

1 INTRODUCTION

Interactive simulation of soft tissue calls for both efficient and physically-plausible methods to model the non-linear deformations. To accomplish this, the finite element method (FEM) is commonly employed. However, the real-time simulation of larger systems with many degrees-of-freedom (DOF) quickly exceeds the capabilities of today's computer hardware. Still, a large number of DOF are necessary to faithfully reproduce the deformations. In this paper, we discuss two strategies to make the simulation of geometrically-complex deformable bodies compatible with the real-time constraints, notably the *adaptive* simulation, and the *embedding* approximation.

In an adaptive simulation of deformable bodies, as *e. g.* discussed by Debunne *et al.* [6], the DOF are dynamically arranged in *regions of interest*, and removed from the undeformed parts. This is particularly attractive in the context of surgery simulations where the interaction between the soft tissue and a surgery tool is considered. Here, the regions of interest are commonly spatially narrow regions around the tip of the tool.

The second strategy, called embedding, is orthogonal to the adaptive simulation. In order to animate a geometrically-complex surface, the surface is embedded into a coarse simulation mesh. The deformation is then governed by the simulation mesh, and the surface vertices are interpolated. Although the embedding technology is known since long [33, 23], it gained increasing attention with the recent advances of Nesme *et al.* that consider the varying material properties of the embedded body in the derivation of the stiffness matrices of the simulation mesh [24]. Still, the unification of an adaptive simulation with the embedding strategy in the context of an interactive simulator is not well investigated.

Contribution In this paper, we propose a method for the adaptive real-time simulation of complex deformable bodies. The key ingredient is a threefold body representation of the body, consisting of a coarse simulation mesh, a mid-resolution tetrahedral collision mesh, and a high-resolution surface for the visualization.

The coarse *simulation mesh* is a non-conforming rectangular grid composed of cubic elements, and governs the deformation and dynamic evolution of the body. The cubic elements allow for an octree representation, and enable the adaptive refinement. The surface-interpolating *tetrahedral mesh* is exclusively employed to detect and resolve collisions. This is because tetrahedra are well-suited for many collision detection schemes. Moreover, the tetrahedra approximate the surface of the body much better than the cubic elements of the simulation mesh, and therefore allow for a precise and physically-plausible handling of boundary conditions. We then present a formulation for transferring the contact displacements back on the simulation mesh such that the momentum is preserved. This contrasts previous approaches that commonly performed the collision handling directly on the surface and therefore degrade the performance if high-resolution surfaces are employed. By employing our threefold body representation, we can

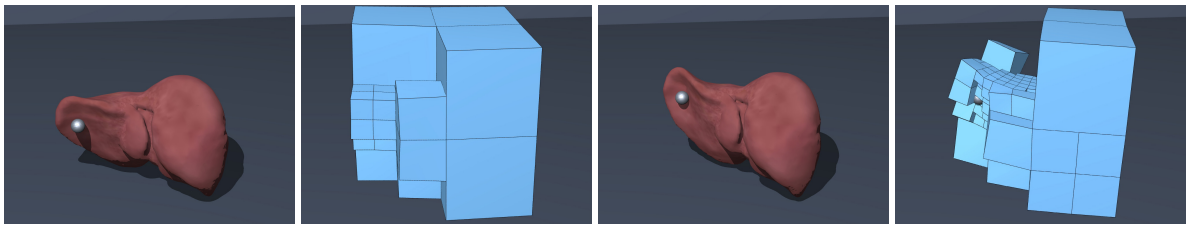

Figure 1: Adaptive resoultion of simulation mesh. A liver model is deformed by a human guided tool. Interactive simulation of a deformable liver. The adaptive resolution allows for an efficient simulation, since the DOF are arranged in the region of interest around the tool. The high-resolution surface is exclusively employed for the visualization.

adaptively simulate complex, highly deformable bodies at interactive rates, as shown in Fig. 1.

2 RELATED WORK

The physically-based simulation of soft bodies is an active area of research. For an overview, we refer to the excellent survey of Nealen *et al.* [25]. In this paper, we focus on the interactive real-time simulation of soft bodies. Due to their simplicity and efficiency, mass-spring systems [2, 33, 11] are frequently used. However, it is not easy to preserve the global deformation behavior of an object under adaptive refinement. The finite element method (FEM) builds directly on the laws of continuum mechanics [13] and is therefore better suited for the adaptive simulation. We employ a linear FE method with a co-rotational stiffness tensor, allowing to reproduce large deformations [17, 18].

Embedding To improve the efficiency, we consider an embedding strategy where a detailed surface mesh is embedded in a coarse simulation mesh [5, 33]. The surface is commonly embedded by linear interpolation, although other techniques such as harmonic coordinates (suitable for arbitrary polygons) [14, 20] or moving least squares [15] have been proposed. Recently, Nesme *et al.* have proposed to compute a kinematic relation between the coarse and the surface nodes by performing a static FE analysis, and then use this relationship for the embedding [24]. We employ linear embeddings since this is favorable in the context of an adaptive simulation.

The usage of cubic elements in the simulation mesh is favorable in the context of adaptive refinement. However, one disadvantage of a hexahedral simulation mesh is that its boundary is not conforming. Therefore, boundary conditions arising from, *e. g.*, contact handling cannot be easily imposed directly on the simulation mesh. Some authors [10, 29, 8] impose the boundary conditions directly on the surface mesh, but this degrades the performance in case of highly detailed surface meshes. Therefore, we propose a novel threefold representation where a low-resolution non-conforming hexahedral mesh governs the deformation, a mid-resolution tetrahedron mesh is employed to detect collisions and impose boundary conditions, and a high-resolution triangle mesh is used for visualization.

Adaptive deformation A widely used strategy to further accelerate the simulation of soft bodies is to employ an adaptive framework. This is usually accomplished by considering a hierarchy of meshes at different resolutions. Then, based on some level-of-detail criterion, each portion of an object can be simulated at a different resolution, and the physical properties are interpolated in between the different resolutions [6]. In case of tetrahedral meshes, the hierarchy is usually precomputed [35, 26] since dynamic splitting and merging of tetrahedra is a difficult task. This is where the use of a hexahedral simulation mesh becomes most evident. This is because the octree-refinement of hexahedrons is particularly simple [7, 23]. In turn, the different resolutions do not need to be pre-computed; instead, the adaptation can be done on the fly. This is particularly impressive demonstrated in [23] where the refinement level is governed by the position and orientation of the camera.

One problem inherent to multi-resolution is the lack of compatibility between elements at different resolutions, resulting in *incompatible nodes* or *T-junctions*. Apart from the strategy of avoiding incompatible elements by, *e. g.*, employing a red-green refinement strategy [16] or extending the model to arbitrary polygons [34, 20], there exist a wealth of approaches to enforce the compatibility explicitly. For example, this can be done by imposing constraints and solving them accordingly, *e. g.*, with the method of Lagrange multipliers [4]. The discontinuous Galerkin FE method (DG-FEM) treats each element in isolation, and employs penalty forces to weakly enforce compatibility [15]. However, it comes with the drawback of simulating more DOF than are actually desired in the simulation. A similar strategy that neglects the deformation of the elements is presented in [3]. As an alternative to these works, hierarchical multi-resolution approaches [9, 23] consider the interaction of an incompatible node with *all* its direct and indirect parents, thereby avoiding a special treatment of the constraints. However, the resulting system tends to be denser, which influences the performance of an implicit solver. We follow the approach of Sifakis *et al.* which propagates the force gradients between incompatible nodes [30]. This approach has the key advantage that

the incompatible nodes do not need to be simulated, and no special constraint handling is necessary.

3 OVERVIEW

We employ a threefold representation to simulate the deformable body. The starting point is a high-resolution surface obtained from, *e. g.*, a geometric modeling process or an MRI scan. From this surface, we compute a tetrahedral mesh whose boundary nodes interpolate the surface. The tetrahedral mesh defines the material distribution of the body. We underline that since the tetrahedral mesh is not employed for the deformation computation, the tetrahedra do not need to be particularly well-shaped.

To dynamically simulate the deformable body, the tetrahedral mesh is embedded into a coarse rectangular simulation mesh whose nodes correspond to Lagrangian mass points with associated dynamic properties. The simulation, including the handling of collisions, is done by employing a standard manifold projection method, which first assembles the global stiffness matrix $\mathbf{K}$, then performs an unconstrained time-evolution to obtain unconstrained positions $\tilde{\mathbf{x}}(t+h)$, and then projects the solution back on the geometric manifold imposed by the contact constraints to obtain non-colliding positions $\mathbf{x}(t+h)$. Since the contact handling is done with the tetrahedral mesh, the positions $\tilde{\mathbf{x}}_c$ of the tetrahedral mesh nodes must be interpolated from the simulated points, and the resulting feasible positions $\mathbf{x}_c$ must be transferred back on the simulation mesh afterwards. As a post-processing step, an oracle, *i. e.* a heuristic decision maker, performs the adaptive refinement in order to arrange the DOF according to the regions of interest.

repeat
  $\mathbf{K} \leftarrow \text{AssembleStiffnessMatrix}(\mathbf{x}(t))$
  $\tilde{\mathbf{x}}(t+h) \leftarrow \text{TimeEvolution}(\mathbf{x}(t), \dot{\mathbf{x}}(t), \mathbf{K})$
  $\tilde{\mathbf{x}}_c \leftarrow \text{UpdateCollisionMesh}(\tilde{\mathbf{x}}(t+h))$
  $\mathbf{x}_c \leftarrow \text{ComputeFeasiblePositions}(\tilde{\mathbf{x}}_c)$
  $\mathbf{x}(t+h) \leftarrow \text{TransferFeasiblePositions}(\mathbf{x}_c)$
  $\text{AdaptiveRefinement}(\mathbf{x}(t+h))$
until stop;

4 SIMULATION MESH

The simulation mesh is a partition of the simulation domain into cubic elements of different sizes. An octree is employed to store the hierarchy of elements. Since we opt for a pure local refinement of elements, we need to handle the resulting mesh incompatibility, *i. e.*, we need to detect and handle the T-junctions. Having in mind that we want to adaptively refine and un-refine the mesh on-the-fly, we first describe a compact representation of the intrinsic coordinates of the simulation elements and nodes. We then describe an approach to detect the T-junctions and discuss the deformation model as well as the numerical integration method.

4.1 Location coordinates

A path to an element of the octree is determined by the information whether we continue on the left or right side of three split planes. This corresponds to a sequence of binary decisions per dimension, with a 0 for left, and a 1 for right. We now store this decisions for each dimension separately and obtain a 3-tuple of bit sequences

$$S_i = \begin{bmatrix} s_x \\ s_y \\ s_z \end{bmatrix} = \begin{bmatrix} x_1 x_2 \ldots \\ y_1 y_2 \ldots \\ z_1 z_2 \ldots \end{bmatrix}$$

where, *e. g.*, $x_1 \in [0,1]$ encodes the information whether the element lies on the left or right side of the top-level plane splitting the X-axis. We denote these 3-tuples as *location coordinate*, similar to the term *location code* [21, 27, 1]. Each component of the location coordinate is now represented in fixed point arithmetic. The main benefit of this representation is that point coordinates can be exactly compared, thus avoiding the headaches that come with floating point comparisons. The navigation within the octree can now be efficiently done by adding appropriate offset vectors.

4.2 Adaptive refinement

By having assigned a unique location coordinate to each element within the octree, we have now the tools in hand to efficiently refining refine elements. We refine an element if the deformation energy of the element is above a certain threshold. This simple heuristic is based on the observation that contacts induce deformations. Then, the DOF must be arranged in order to reproduce the deformation, as we will show later. If, in contrast, the deformation energy of a refined element is below a second threshold, then we merge its sub-elements. It is important not to merge strongly deformed elements, since this results in popping artifacts [26].

To refine, we recursively split an element into eight sub-elements of same size. Further, we underline that the refinement is purely local, without restriction on the level difference between adjacent elements, which is in contrast to[7]. This avoids that refinements are propagated throughout the entire mesh, which would degrade the performance.

However, at this point we have to consider that the original simulation mesh does not correspond to the simulated body, but to the cubic bounding-box of the *embedded* body. Therefore, after a refinement, some elements might not contain any material (see Fig. **??**). These void elements do not need to be simulated. Thus, after having refined an element, we test each sub-element for material intersection, *i. e.*, we determine the intersection of the sub-elements with the tetrahedral mesh and remove the empty elements from the simulation.

Following [6], we interpolate the positions and velocities linearly. In order to adjust the mass contribu-

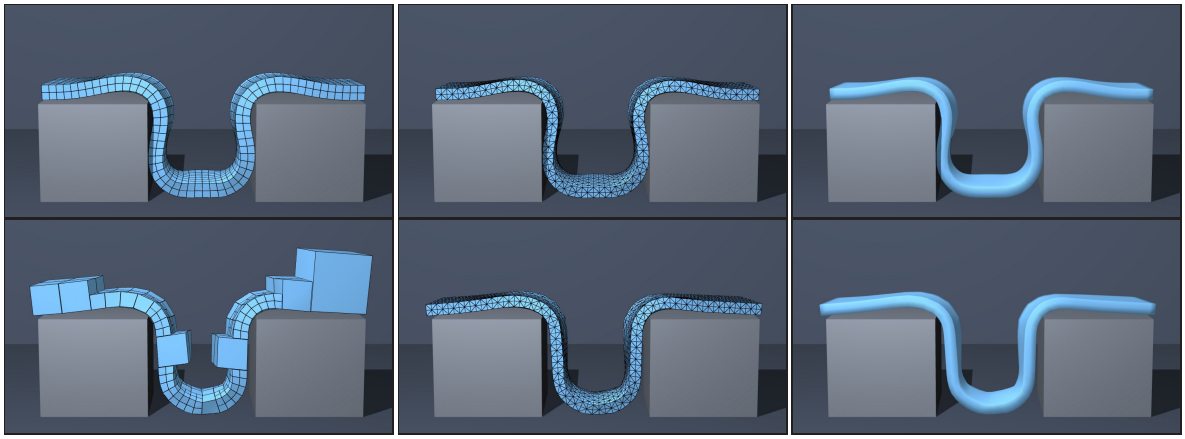

Figure 2: We employ a threefold representation of the body, notably a deformation mesh, a collision mesh, and an embedded surface. The *top row* shows the result of a full simulation while *bottom row* shows the same scene but with an adaptive simulation, where elements are refined in regions of large deformation.

tions of the sub-elements, we iterate over the embedded tetrahedral corner points and distribute their masses barycentrically on the simulation nodes. In addition, we scale the stiffness matrix in order to account for elements that are only partially filled with material. As Nesme *et al.* point out, this improves the stability, because lighter elements are now also softer [23].

4.3 Detecting T-junctions

T-junctions are inevitable if elements in an octree are refined in isolation, *i. e.*, without considering the neighborhood of the element. While refining elements in isolation is favorable with respect to efficiency and implementation ease, the T-junctions require a special treatment in order to avoid cracks during the simulation. To check whether a given node is a T-junction, we traverse its set of at most eight adjacent elements, and check whether the element is not empty *and* the node is not a corner of that element. Since T-junctions are not known a priori, this has to be done for each node. Further, since the height of the unbalanced octree is in $\mathcal{O}(N)$, the T-junction detection is in $\mathcal{O}(N^2)$ in theory. Still, since numerical issues forbid an unlimited refinement, the height of the tree can be considered as constant in practice. By imposing restrictions on the level differences, linear-time T-junction detection algorithms can be designed [7, 1], but this comes at the price of a non-local refinement.

4.4 Dynamic deformation model

The adaptive refinement of the simulation mesh results in a complex of cubic elements and mass points. To compute the elastic response of the deformable body, we rely on the linear co-rotational FE method as discussed in, *e. g.*, [18]. This approach has the advantage that for each element, the stiffness tensor $\mathbf{K}_e$ depends solely on the un-deformed point coordinates and an element rotation $\mathbf{R}_e$. Since in the element frames, the elements are axis-aligned cubes, we can pre-compute the elemental stiffness tensors in an analytical way, see Appendix. To estimate the rotation, we follow the approach described in [19]. We then assemble a global stiffness matrix $\mathbf{K}$ from the elemental stiffness matrices $\mathbf{K}_e$.

To compute the dynamic evolution of the deformable body, we assume that the mass points obey the Newton equations of motion. The time-dependent positions and velocities of the points then result from numerically integrating the equations of motion with a semi-implicit solver. We use the pre-conditioned conjugate gradient from the Taucs library[1] to solve the linear system.

4.5 T-junction handling

At this point, we have considered all nodes of the simulation mesh as DOF. However, this is not true for the T-junction nodes which are barycentrically constrained to edges or faces of adjacent elements. To enforce these constraints and avoid cracks in the simulation, we follow an approach proposed by Sifakis *et al.* in [30] and later generalized by Nesme *et al.* [24]. The key observation is that there exists a linear relation between the coordinates $\mathbf{x}_{full}$ of the full system, and the coordinates $\mathbf{x}_{redu}$ of the reduced system containing only real DOF,

$$\mathbf{x}_{full} = \mathbf{W}\mathbf{x}_{redu} \tag{1}$$

where the matrix $\mathbf{W}$ contains the barycentric coordinates of the T-junctions with respect to their parents [30]. Nesme *et al.* have shown that a similar relation holds between the nodal forces,

$$\mathbf{f}_{redu} = \mathbf{W}^T\mathbf{f}_{full} \tag{2}$$

where the vector $\mathbf{f}_{full}$ contains the forces on *all* points [24]. The operator $\mathbf{W}^T$ distributes these onto the reduced points, excluding the hard bound points. Since

[1] http://www.tau.ac.il/~stoledo/taucs/

we employ an implicit solver, we assemble the global stiffness matrix likewise as

$$\mathbf{K}_{redu} = \mathbf{W}^T \mathbf{K}_{full} \mathbf{W}. \quad (3)$$

We now employ the reduced stiffness matrix $\mathbf{K}_{redu}$ to solve for the future positions. The advantage of this approach is that the dimension of the system is reduced by the number of T-junctions, and that no special constraint treatment is required. The downside, however, is that assembling the reduced system is more expensive, and that the reduced system is denser.

5 TETRAHEDRAL MESH

While the simulation mesh governs the deformation and dynamic evolution of the body, boundary conditions such as contact constraints are imposed on the tetrahedral mesh. In this section, we describe how to update the tetrahedral mesh, handle collisions, and transfer the displacements back onto the simulation mesh.

5.1 Mesh update

We assume that each node of the tetrahedral mesh lies in exactly one element of the simulation mesh. Thus, we can derive a linear relation between the simulation mesh nodes $\mathbf{x}$ and the tetrahedral mesh nodes $\mathbf{x}_c$,

$$\mathbf{x}_c = \mathbf{H}\mathbf{x} \quad (4)$$

where the matrix $\mathbf{H}$ contains the barycentric coordinates of the tetrahedral mesh nodes with respect to the simulation mesh nodes. These correspond to the linear shape functions evaluated at $\mathbf{x}_c$. The matrix $\mathbf{H}$ needs to be recomputed only after an adaptive refinement.

Consequently, having computed the (colliding) future simulation mesh positions $\tilde{\mathbf{x}}(t+h)$, we update the tetrahedral mesh nodes with $\tilde{\mathbf{x}}_c(t+h) = \mathbf{H}\tilde{\mathbf{x}}(t+h)$. Then, the collision handling is performed on the (colliding) positions $\tilde{\mathbf{x}}_c$, as described in the following.

5.2 Collision handling

To detect inter-object collisions, we employ a spatial hashing approach [32]. Then, the penetration depths are computed with the approach described in [12]. Since both approaches rely on a tetrahedral discretization, they go hand in hand with our threefold representation.

To resolve collisions, we employ a position-based scheme similar in spirit to [28] and [31]. That is, we resolve each point-triangle collision locally while conserving the momentum. This process is repeated iteratively until all penetrations are resolved. Experiments indicate that 5-10 iterations provide sufficient accuracy. The result of the computation are collision-free positions $\mathbf{x}_c$ and corresponding displacements $\mathbf{d}_c = \mathbf{x}_c - \tilde{\mathbf{x}}_c$.

5.3 Transferring the displacements

Having computed the collision displacements, we have to transfer the solution back onto the simulation mesh in order to realize the collision response. Still, this has to be accomplished such that the momentum of the simulation mesh is as well preserved. Moreover, one has to consider that the resolutions of the simulation mesh and the tetrahedral mesh might differ significantly.

To accomplish this, we again resort to [24] which relates the forces $\mathbf{f}$ on the coarse nodes to the forces $\mathbf{f}_c$ on the embedded points as

$$\mathbf{f} = \mathbf{H}^T \mathbf{f}_c. \quad (5)$$

To apply this relation, we write $\mathbf{f}_c = \frac{1}{2}h^{-2}\mathbf{M}_c\mathbf{d}_c$, where $\mathbf{M}_c$ is the diagonal mass matrix of the tetrahedral mesh. Then, $\mathbf{f} = \frac{1}{2}h^{-2}\mathbf{H}^T\mathbf{M}_c\mathbf{d}_c$. Still, it is favorable to handle collisions on the velocity level. Thus, we convert the resulting contact forces $\mathbf{f}$ back to contact displacements $\mathbf{d}$, and obtain the final displacement transfer formulation as

$$\mathbf{d} = 2h^2\mathbf{M}^{-1}\left(\frac{1}{2}h^{-2}\mathbf{H}^T\mathbf{M}_c\mathbf{d}_c\right) = \mathbf{M}^{-1}\mathbf{H}^T\mathbf{M}_c\mathbf{d}_c \quad (6)$$

where $\mathbf{M}$ is the diagonal mass matrix of the simulation mesh nodes. We then update the positions with $\mathbf{x}(t+h) \leftarrow \tilde{\mathbf{x}}(t+h) + \mathbf{d}$, and the velocities with $\dot{\mathbf{x}}(t+h) \leftarrow h^{-1}(\mathbf{x}(t+h) - \mathbf{x}(t))$. The key benefit of this formulation is that the momentum is preserved, which is crucial for the physical plausibility of the simulation.

To understand the consequence of this transfer formulation, one has to consider the case where many tetrahedral points are embedded in a single element. In this case, the masses of the simulation nodes are larger than the masses of the collision nodes. Consequently, the resulting simulation node displacements $\mathbf{d}$ are smaller than the computed displacements $\mathbf{d}_c$, and the collisions cannot be resolved. This problem can only be overcome by dynamically refining the coarse elements in the region of the interaction. In turn, this results in smaller and lighter elements that can properly react to the collision. Therefore, a good adaptation oracle is crucial.

6 SURFACE MESH

In order to improve the fidelity, we employ a high-resolution triangle surface for the visualization. This surface is linearly embedded in the tetrahedral mesh, having the advantage that the barycentric coordinates do not need to be re-computed after refinement of the simulation mesh. Another benefit is that since we compute the non-colliding positions of the tetrahedral nodes and update the vertices of the surface mesh thereafter, visible collisions are avoided even if the simulation mesh fails to reproduce the deformation accurately enough, as discussed before.

Scene		High-res	Adaptive
Bar	element count	1024	265
	ms / time step	120	17
	speed-up factor	–	7x
Plane	element count	896	75
	ms / time step	102	8
	speed-up factor	–	13x

Table 1: Performance measurements of our method. The first row shows the timings of the adaptive bar simulation illustrated in Fig. 2. For this scenarios, we obtain a speed-up a factor of 7 by employing the adaptive model. The second row illustrates that for the interactive plane scene (see Fig. 4), we obtain a speed-up of a factor of 13 since we only refine the mesh in the region of interest around the tool.

7 RESULTS

In this section, we evaluate the performance of our method, and demonstrate our method in various challenging interactive and off-line scenarios. We underline that the pre-computation time of our method is negligible. All experiments have been performed on an Intel quad-core CPU @ 2.6 GHz with a GeForce GTX 280.

7.1 Performance

To show that the application of our method results in a significant performance gain without compromising the accuracy, we perform an experiment where an elastic bar is laid between two rigid obstacles (see Fig. 2). The length of the bar is 0.2m, its density is 2×10^5 kg m^{-3}, the Young modulus is 5×10^5 Pa, and the Poisson ratio is 0.4. We perform the experiment with a non-adaptive high resolution simulation mesh and compare it with an adaptive simulation mesh. The measurements (see Tab. 1) indicate a speed-up factor of seven with only minor loss in visual quality.

7.2 Examples

First we verify that our method yields comparable results as a reference simulation: we use tetrahedral FEM to simulate the same body at a higher resolution. Fig. 3 shows for one example that this is indeed the case.

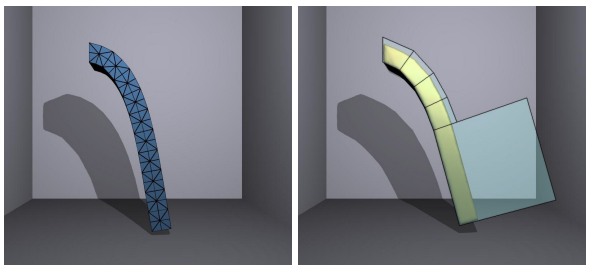

Figure 3: Qualitative comparison of a hanging bar. Left: The deformation mesh of the tetrahedral FEM consists out of 800 elements. Right: Our method embeds the body into 9 cubic elements.

Our method is particularly attractive in scenarios where a user interacts in a spatially coherent area of a large object. To illustrate this, we perform an experiment where the user interacts with an elastic plane lying on the ground (see Fig. 4). The size of the plane is 0.2m $\times$ 0.2m, its density is 400kg m^{-3}, the Young modulus is 3×10^4 Pa, and the Poisson ratio is 0.4. Since we employ a co-rotational stiffness tensor, large deformations can be reproduced plausibly, as illustrated in Fig. 4. There are on average 75 elements simulated, and the tetrahedral collision mesh contains 8k elements. The average computation of one simulation step takes 8ms, where 10^{-3}ms are spent for the adaptive (un-)refinement. The simulation runs at 19 frames per second on average. A similar performance cannot be achieved by employing a non-adaptive model (see Tab. 1). In this case, the frame rate drops to 3 fps, and an interactive manipulation is not possible.

To show that our method can also be employed for a complex off-line simulation, we drop 81 elastic toy bikes discretized into 384 tetrahedra on the ground (see Fig. 5). The computation of the dynamics of up to 3k elements takes 70ms per time step on average. The detection of 220 collisions takes 20ms and the computation of the non-colliding positions, including transferring the displacements, takes 7ms. The surface meshes of the bikes contain 13k triangles each, performing the collision handling on the surfaces directly would therefore result in a significant computational overhead. In total, the simulation takes about 7s per frame, including the rendering of 2.5 million surface triangles.

8 CONCLUSION

In this paper, we have presented an approach for robust and efficient simulation of deformable bodies. The key ingredient has been a novel threefold representation, where a coarse simulation mesh governs the deformation and dynamic evolution, a tetrahedral mesh is employed to handle collisions, and a high-resolution surface is visualized. The simulation mesh is a partition of the domain into axis-aligned cubes, and therefore enables a straight-forward adaptive refinement. We have then discussed an efficient approach to detect the T-junctions. A linear co-rotational FE method, in tandem with an implicit numerical integration method, is employed to compute the dynamic evolution.

We have further discussed a physically-plausible way to update the embedded tetrahedral mesh, and afterwards transfer the collision displacements back onto the deformation mesh. Our formulation preserves the momentum and is therefore physically plausible. Experiments indicate that we can simulate interacting deformable bodies at interactive rates.

As future work, we plan to adapt the stiffness tensor in order to reflect the embedded material, similar in spirit to Nesme *et al.* [24]. In addition, we will address topological changes such as cutting and fracture.

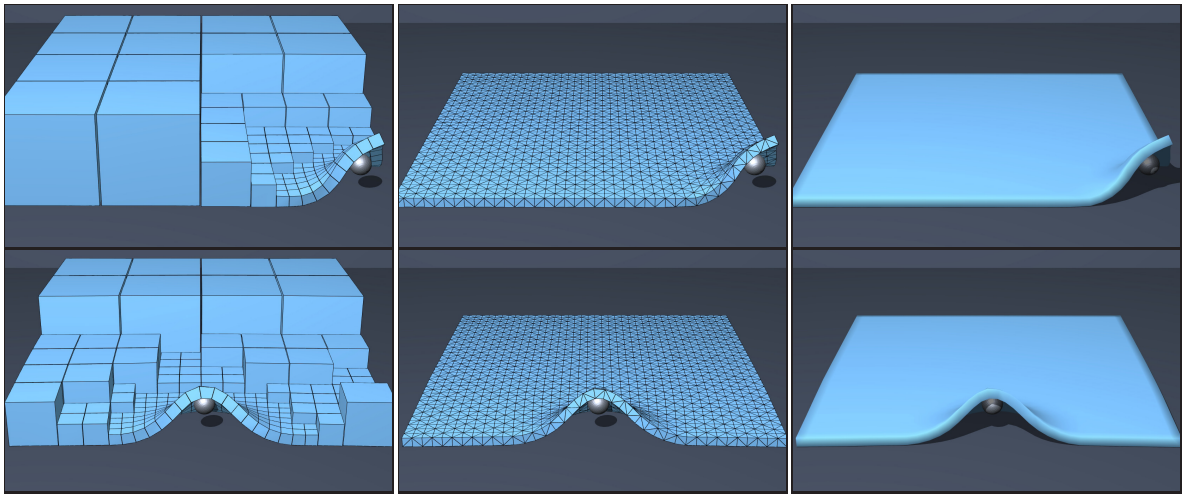

Figure 4: By employing an adaptive scheme, we can reproduce rich deformations at interactive rates. This is because a large object is refined locally, thus arranging the DOF in the regions of interest. To illustrate this, we perform a simulation of a deformable plane being manipulated by a user. The simulation runs at 19 frames per second on average. Left: The hexahedral simulation grid. Middle: The tetrahedral collision mesh. Right: The high-resolution surface mesh.

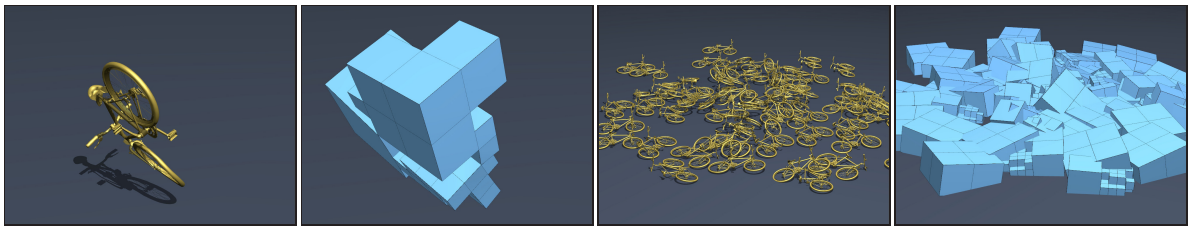

Figure 5: Massive scenario of bikes falling on the ground. The collisions induce deformations, which in turn make that the objects are being refined. The surface meshes of the bikes contain 13k triangles each. Since we perform the collision handling on the tetrahedral mesh, the simulation of the full scenario depicted on the right takes only 7s per frame.

This work has been supported by the Swiss CTI project ArthroS.

REFERENCES

[1] Aizawa K., Tanaka S.: A constant-time algorithm for finding neighbors in quadtrees. vol. 31, IEEE Computer Society, pp. 1178–1183.

[2] Bridson R., Fedkiw R., Anderson J.: Robust treatment of collisions, contact and friction for cloth animation. ACM Transactions on Graphics (2002), 594–603.

[3] Botsch M., Pauly M., Wicke M., Gross M.: Adaptive space deformations based on rigid cells. Computer Graphics Forum 26, 3 (2007), 339–347.

[4] Carey G. F.: Computational Grids: Generation, Adaptation and Solution Strategies. Taylor & Francis, 1997.

[5] Capell S., Green S., Curless B., Duchamp T., Popović Z.: Interactive skeleton-driven dynamic deformations. ACM Transaction on Graphics (Proc. SIGGRAPH) 21, 3 (2002), 586–593.

[6] Debunne G., Desbrun M., Cani M.-P., Barr A. H.: Dynamic real-time deformations using space and time adaptive sampling. in Proc. SIGGRAPH (2001), pp. 31–36.

[7] Dequidt J., Marchal D., Grisoni L.: Time-critical animation of deformable solids: Collision detection and deformable objects. Computer Animation and Virtual Worlds 16, 3-4 (2005), 177–187.

[8] Faure F., Barbier S., Allard J., Falipou F.: Image-based collision detection and response between arbitrary volumetric objects. in ACM Siggraph/Eurographics Symposium on Computer Animation, SCA 2008, July, 2008 (2008).

[9] Grinspun E., Krysl P., Schröder P.: CHARMS: a simple framework for adaptive simulation. ACM Transactions on Graphics (Proc. SIGGRAPH) 21, 3 (2002), 281–290.

[10] Galoppo N., Otaduy M. A., Tekin S., Gross M., Lin M. C.: Soft articulated characters with fast contact handling. Computer Graphics Forum 26, 3 (2007), 243–253.

[11] Georgii J., Westermann R.: Mass-spring systems on the gpu. Elsevier Science (July 2005).

[12] Heidelberger B., Teschner M., Keiser R., Mueller M., Gross M.: Consistent penetration depth estimation for deformable collision response. in Proc. Vision, Modeling, Visualization (2004), pp. 339–346.

[13] Hughes T.: The Finite Element Method. Prentice-

Hall, NJ, 1987.

[14] Joshi P., Meyer M., DeRose T., Green B., Sanocki T.: Harmonic coordinates for character articulation. ACM Trans. Graph. 26, 3 (2007), 71.

[15] Kaufmann P., Martin S., Botsch M., Gross M. H.: Flexible simulation of deformable models using discontinuous galerkin fem. Graphical Models 71, 4 (2009), 153–167.

[16] Molino N., Bridson R., Teran J., Fedkiw R.: A crystalline, red green strategy for meshing highly deformable objects with tetrahedra. in Proc. 12th International Meshing Roundtable (2003), pp. 103–114.

[17] Müller M., Dorsey J., McMillan L., Jagnow R., Cutler B.: Stable real-time deformations. in Proc. ACM SIGGRAPH/Eurographics symposium on Computer animation (2002), pp. 49–54.

[18] Müller M., Gross M.: Interactive virtual materials. in Proc. Graphics Interface (2004), pp. 239–246.

[19] Müller M., Heidelberger B., Teschner M., Gross M.: Meshless deformations based on shape matching. ACM Transaction on Graphics (Proc. SIGGRAPH) 24, 3 (2005), 471–478.

[20] Martin S., Kaufmann P., Botsch M., Wicke M., Gross M. H.: Polyhedral finite elements using harmonic basis functions. Comput. Graph. Forum 27, 5 (2008), 1521–1529.

[21] Morton G.: A computer oriented geodetic data base and a new technique in file sequencing, 1966.

[22] Muller M., Teschner M., Gross M.: Physically-based simulation of objects represented by surface meshes. in CGI '04: Proceedings of the Computer Graphics International (2004), pp. 26–33.

[23] Nesme M., Faure F., Payan Y.: Hierarchical multi-resolution finite element model for soft body simulation. in 2nd Workshop on Computer Assisted Diagnosis and Surgery, March, 2006 (2006).

[24] Nesme M., Kry P. G., Jerábková L., Faure F.: Preserving topology and elasticity for embedded deformable models. in SIGGRAPH '09: ACM SIGGRAPH 2009 papers (2009).

[25] Nealen A., Müller M., Keiser R., Boxermann E., Carlson M.: Physically Based Deformable Models in Computer Graphics. in Eurographics-STAR (2005), pp. 71–94.

[26] Otaduy M., Germann D., Redon S., Gross M.: Adaptive deformations with fast tight bounds. in Proc. ACM SIGGRAPH/Eurographics Symposium on Computer Animation (2007), pp. 181–190.

[27] Samet H.: The Design and Analysis of Spatial Data Structures (Addison-Wesley series in computer science). Addison-Wesley Pub (Sd), 1989.

[28] Spillmann J., Becker M., Teschner M.: Non-iterative computation of contact forces for deformable objects. Journal of WSCG 15, 1-3 (2007), 33–40.

[29] Steinemann D., Otaduy M. A., Gross M.: Efficient bounds for point-based animations. in Proc. IEEE/Eurographics Symposium on Point-Based Graphics (2007), pp. 57–64.

[30] Sifakis E., Shinar T., Irving G., Fedkiw R.: Hybrid simulation of deformable solids. in Proc. ACM SIGGRAPH/Eurographics symposium on Computer animation (2007), pp. 81–90.

[31] Spillmann J., Teschner M.: An Adaptive Contact Model for the Robust Simulation of Knots. Computer Graphics Forum (Proc. Eurographics) 27, 2 (2008), 497–506.

[32] Teschner M., Heidelberger B., Müller M., Pomerantes D., Gross M. H.: Optimized spatial hashing for collision detection of deformable objects. in Proc. Vision, Modeling, Visualization (2003), pp. 47–54.

[33] Teschner M., Heidelberger B., Muller M., Gross M.: A versatile and robust model for geometrically complex deformable solids. in Proc. Computer Graphics International (2004), pp. 312–319.

[34] Wicke M., Botsch M., Gross M.: A finite element method on convex polyhedra. Computer Graphics Forum (Proc. Eurographics) 26, 3 (2007), 355–364.

[35] Wu X., Downes M., Goktekin T., Tendick F.: Adaptive nonlinear finite elements for deformable body simulation using dynamic progressive meshes. Computer Graphics Forum 20, 10 (2001), 349–358.

A CUBIC ELEMENT STIFFNESS

In this section, we discuss a way to analytically compute the element stiffness tensor $\mathbf{K}_e$, similar in spirit to [22]. For general hexahedra the usual approach is to use the isoparametric approach and perform a numerical integration of the resulting integral. For axis-aligned cubes this is not necessary and a simple parametrization of the integration domain is straight-forward. From continuum mechanics we know that a body's deformation energy is given by

$$E_{def} = \int_\Omega \frac{1}{2}\sigma^T \varepsilon dV = \int_{z_1}^{z_2}\int_{y_1}^{y_2}\int_{x_1}^{x_2} \frac{1}{2}\varepsilon^T \mathbf{D}\varepsilon \, dxdydz \qquad (7)$$

with the strain ε and the linear elasticity matrix $\mathbf{D}$. To interpolate the deformation within the element, we employ *natural coordinates*

$$N_i(\xi,\eta,\chi) = \frac{1}{8}(1+\xi_i\xi)(1+\eta_i\eta)(1+\chi_i\chi) \qquad (8)$$

where ξ_i, η_i, and χ_i denote the positions of the corners in natural coordinates. Then the deformation field $\mathbf{u}$ within the element is interpolated by $\mathbf{u}(\xi,\eta,\chi) = \sum_{i=1}^{8} N_i(\xi,\eta,\chi)\mathbf{u}_i$. The Cauchy strain $\varepsilon_c(\mathbf{u}) := \frac{1}{2}(\nabla\mathbf{u} + (\nabla\mathbf{u})^T)$ depends on the spatial derivative

$$\nabla\mathbf{u} = [\frac{\partial\mathbf{u}}{\partial x}, \frac{\partial\mathbf{u}}{\partial y}, \frac{\partial\mathbf{u}}{\partial z}]^T = \mathbf{J}^{-1}[\frac{\partial\mathbf{u}}{\partial\xi}, \frac{\partial\mathbf{u}}{\partial\eta}, \frac{\partial\mathbf{u}}{\partial\chi}]^T \qquad (9)$$

where

$$\mathbf{J} = \begin{pmatrix} \frac{\partial x}{\partial\xi} & \frac{\partial y}{\partial\xi} & \frac{\partial z}{\partial\xi} \\ \frac{\partial x}{\partial\eta} & \frac{\partial y}{\partial\eta} & \frac{\partial z}{\partial\eta} \\ \frac{\partial x}{\partial\chi} & \frac{\partial y}{\partial\chi} & \frac{\partial z}{\partial\chi} \end{pmatrix} \qquad (10)$$

is the Jacobian matrix. For a general hexahedron, x, y and z are functions of ξ, η and χ, and therefore the partial derivatives are involved functions in the integration coordinates, requiring a numerical evaluation of the integral in (7). However, since we have cubes as integration elements, we obtain much simpler functions

$$x(\xi) = 1/2(x_2 - x_1)\xi \qquad (11)$$

$$y(\eta) = 1/2(y_2 - y_1)\eta \qquad (12)$$

$$z(\chi) = 1/2(z_2 - z_1)\chi \qquad (13)$$

with a corresponding constant and diagonal Jacobian $\mathbf{J}$. Thus the integral (7) can be evaluated analytically. The corresponding stiffness tensor results from basic FE theory [13].