Workshop on Virtual Reality Interaction and Physical Simulation VRIPHYS (2010)
J. Bender, K. Erleben, and M. Teschner (Editors)

Interactive Deformations with Multigrid Skeletal Constraints

J. Georgii[†1,2], D. Lagler[1], C. Dick[1], and R. Westermann[1]

[1] Computer Graphics & Visualization Group, Technische Universität München, Germany
[2] Fraunhofer MEVIS, Bremen, Germany

Abstract

In this paper we present an interactive method for simulating deformable objects using skeletal constraints. We introduce a two-way coupling of a finite element model and a skeleton that is attached to this model. The skeleton pose is determined via inverse kinematics. The target positions of joints are either given by user interactions or forces imposed by the surrounding deformable body. The movement of the deformable body either follows the movement of the skeleton thereby respecting physical constraints imposed by the underlying deformation model, or the movement is determined from user-defined external forces. Due to the proposed two-way coupling, the skeleton and the deformable body constrain each other's movement, thus allowing for an intuitive and realistic animation of soft bodies. To realize the two-way coupling we propose the efficient embedding of the constraints into a geometric multigrid scheme to solve the governing equations of deformable body motion. We present a greedy approach that propagates the constraints to coarser hierarchy levels, and we show that this approach can significantly improve the convergence rate of the multigrid solver.

Categories and Subject Descriptors (according to ACM CCS): I.3.5 [Computer Graphics]: Computational Geometry and Object Modeling—Physics-based modeling I.3.7 [Computer Graphics]: Three-Dimensional Graphics and Realism—Animation

1. Introduction

In computer graphics, physics-based simulation of deformable objects has been studied for several decades, starting with the seminal work of Terzopoulos et al. [TPBF87, TF88]. A good overview of the state-of-the-art in this field can be found in [NMK*05]. Besides the development of ever improving techniques for the efficient, yet accurate simulation of deformable bodies, a separate field has evolved dealing with the *intuitive control* of such bodies. Especially for animating virtual characters composed of deformable parts, control mechanisms that respect both the physical properties of these parts and the pose of the character are required.

Early work in this field was done by Isaacs and Cohen, who proposed a set of controls for dynamic simulations including kinematic constraints and inverse dynamics [IC87]. Gourret et al. proposed an approach for the simulation of human skin in a grasping task [GTT89], which is based on a one-way coupling of the skeletal movement to the soft tissue. This idea was later extended by Cappell et al. to allow for pose-based simulation of deformable bodies [CGC*02a]. They transfer the movements of an animated skeleton—obtained via motion capturing—to a volumetric finite element model. Analogously to skinning, the piece-wise linear deformations induced by the skeleton are blended to achieve smooth deformations of the character's surface. However due to this, especially locally the method can result in non-realistic deformations.

To simulate specific surface deformations caused by actions such as breathing, force templates were introduced in [CBC*05]. The authors propose a non-linear optimization technique that computes skeletal states and force fields from a given deformed surface to achieve an accurate matching. Song et al. [SZHB06] simulated the skeleton as a non-linear finite element model, and they propagated the resulting element rotations to the surrounding body to enable (corotated) linear elastic simulation. More realistic simulations of human anatomy were achieved by incorporat-

† georgii@tum.de

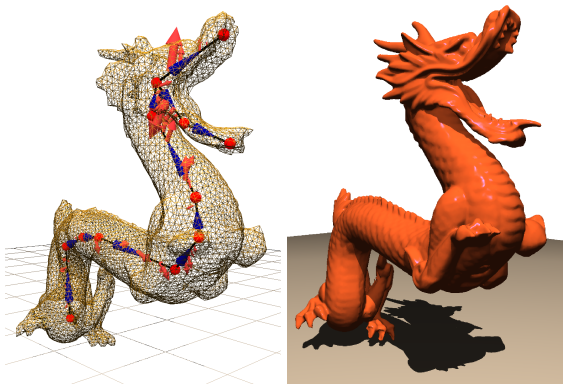

Figure 1: *Left: The skeleton and the forces acting on it (red arrows). Right: An elastic body discretized into 67K tetrahedral finite elements is deformed at a rate of 5 time steps per second using a two-way coupling between the skeleton and the body.*

ing anatomical structures like muscles into the simulation [SPCM97, WVG97].

As soon as such virtual characters interact with their environment, e.g. colliding with other objects, a fully two-way coupling of the deformable body and its control skeleton is required. This enables that the character's control skeleton reacts on collisions that are detected by the surrounding soft body, and the movement of the soft body is fully constrained by the skeleton. Shinar et al. addressed these issues by simulating the skeleton as articulated rigid bodies, and they coupled this skeleton simulation to the soft body using an uniform time integration approach [SSF08], thereby allowing the creation of life-like creatures. However, using a rigid body approach to simulate the skeleton requires a carefully adapted time integration approach to achieve stable simulations [SSF08]. For that reason, we decided to use an inverse kinematic approach to simulate the poses of the control skeleton, which can run with the same simulation rate as the soft body simulation at low computational costs. Moreover, this allows us to easily define constraints in the movements of the skeleton, i.e. by specifying joint limits.

1.1. Our Contributions

We extend previous work in that we propose a physics-based *two-way coupling* of a skeleton and a deformable body to provide an intuitive control mechanism for deformable body animation (see Figure 1). Due to this coupling, any deformation that is transferred from the skeleton to the body or vice versa results in an immediate response of the respective other part, which, in turn, affects the movement of the driving part. Thus, the user can control either the skeleton or the deformable parts surrounding this skeleton. Since the pose of the skeleton is updated automatically due to the forces induced on the skeleton by the deformable body, our approach can be used to determine the skeletal pose automatically from issued volume deformations (see Figure 2 for an example).

We describe the efficient embedding of the aforementioned two-way coupling into a geometric multigrid approach. One direction of this coupling is to transfer the movement from the skeleton to the soft body. We realize this by incorporating displacement boundary conditions into the soft body simulation. The other direction of the coupling is realized by determining the forces acting on the boundary vertices, and by using these forces to update the skeletal pose (thereby respecting the mass of the underlying soft body). Our paper describes a novel approach to handle such displacement boundary conditions in a geometric multigrid scheme by propagating the boundary conditions to coarser levels if necessary. Specifically we introduce a greedy approach that ensures the resulting interpolation and restriction operators to have full rank. This method can be used for nested as well as non-nested geometric grid hierarchies [GW06]. As a result, our multigrid solver achieves good convergence rates and enables interactive update rates even for high-resolution models.

2. Related Work

In computer graphics, deformable models were first introduced by Terzopolous et al. [TPBF87, TF88]. A good overview of the multitude of methods for realistically simulating deformable bodies can be found in [NMK*05]. For example, boundary element models [JP99], adaptive and multiresolution approaches [DDCB01, CGC*02b, GKS02], gridless techniques [MKN*04, MHTG05, Mül08], and finite element methods [BNC96, WDGT01] have been proposed. Nesme et al. [NKJF09] proposed a composite element formulation that considers varying material properties within a coarse element.

Multigrid approaches [Bra77, TOS01] for the solution of large linear systems have recently gained much attention in the computer graphics community due to their linear time complexity. Applications range from fluid simulation [BFGS03] over deformable body simulation [WT04, GW06, SYBF06, ZSTB10] to image processing [KH08]. The efficiency of multigrid methods on hexahedral grids has been shown in [DGBW08].

A general framework for handling constraints in linear time is given in [Bar96]. Bender showed the advantages of impulse-based dynamics to solve the constraints in linear time [Ben07]. Bridson et al. developed an approach to robustly process collisions, contact and friction in cloth simulation [BFA02]. Animation techniques for human athletics are described in [HWBO95]. Weinstein et al. proposed a post-stabilization approach for the simulation of articulated rigid bodies [WTF06], and they later extended it to control joints and muscles [WGF08]. Virtual actors with life-like

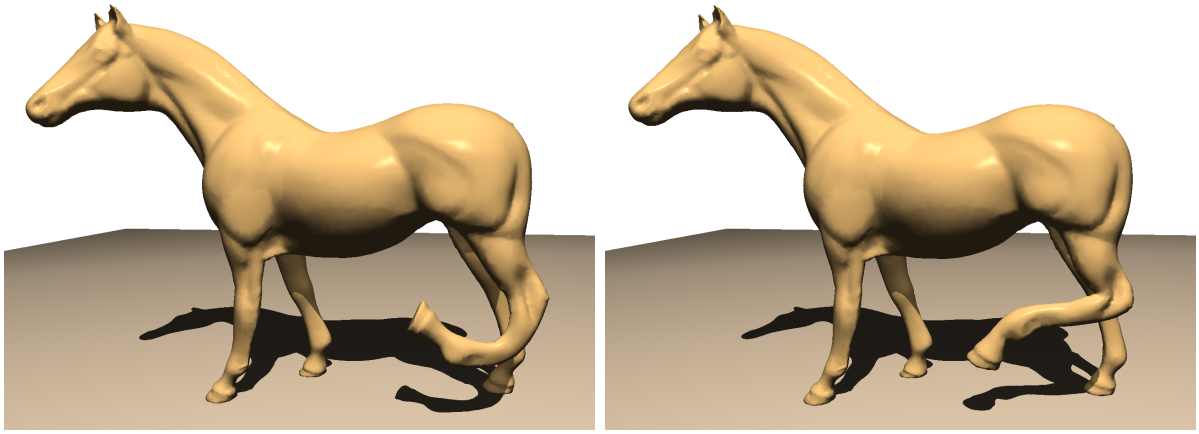

Figure 2: *Deformation of the model of a horse when moving the hoof. Without skeletal constraints an unrealistic deformation is achieved (left). Skeletal constraints provide the desired behavior (right).*

motor skills are described in [FvdPT01]. The interplay between the simulation of a deformable surface and a skeleton is described in [GOT*07].

There is also a vast body of literature addressing the problem to bind a surface to a movable skeleton, which is commonly referred to as skinning, and a comprehensive review is beyond the scope of this paper. However, Singh et al. [SF98], Allen et al. [ACP02], and Ju et al. [JZvdP*08] discuss some of the common techniques used and provide many useful references on this topic.

3. Finite Element Simulation

For the numerical simulation of the dynamic behavior of an elastic solid under the influence of external forces we employ a finite element discretization of the solid using tetrahedral or hexahedral elements. We use an implicit multigrid solver as proposed in [GW06, GW08], which provides built-in support for corotated elements [MDM*02, HS04, MG04] to enable large deformations. In particular, a data structure that is specifically designed to support efficient sparse matrix products is employed for updating the multigrid hierarchy [GW10]. In every time step a system of equations $Ku = f$ needs to be solved, which results from the time integration of the Lagrangian equation of motion. Here, u is the linearized displacement vector and f contains the forces applied to the finite element vertices. An implicit second-order Newmark scheme is used for time integration.

3.1. Displacement Boundary Conditions

The deformation of an elastic body is typically performed by specifying an appropriate force vector—or a force field—that acts on the finite elements and results in a displacement of element vertices. However, if the deformation is issued by a skeleton attached to the body, one expects that the element vertices on the skeleton are constrained to the skeleton (displacement boundary conditions) and only for the remaining vertices the displacements are simulated. Since it becomes very difficult to accurately model the displacement boundary conditions by corresponding forces, for the set of vertices on the skeleton displacements should be directly specified instead of external forces.

By assuming that a displacement u_i or a force f_i is given for every vertex v_i, the entire set of vertices can be grouped into two subsets S_1 and S_2: The first subset contains all vertices for which a displacement boundary condition has to be enforced and the respective force needs to be computed, and the second subset contains all vertices for which a force is given (for instance, due to user interaction or gravity) and the respective displacement has to be computed. We can thus split the linearized displacement and force vectors into u_1, f_1 and u_2, f_2, where u_1 and f_2 are known and f_1 and u_2 have to be computed. The system of equations to be solved in every simulation step can then be partitioned as

$$\begin{pmatrix} K_{11} & K_{12} \\ K_{21} & K_{22} \end{pmatrix} \begin{pmatrix} u_1 \\ u_2 \end{pmatrix} = \begin{pmatrix} f_1 \\ f_2 \end{pmatrix}.$$

This system can be solved by first solving the (typically) smaller system $K_{22}u_2 = f_2 - K_{21}u_1$ for the unknown displacements u_2 of all vertices $v \in S_2$ and then using these displacements to determine the unknown forces exerting at vertices $v \in S_1$. However, to enable the use of efficient geometric multigrid methods to solve the system, one cannot easily stick with the smaller system since it removes degrees of freedom of the system, which corresponds to "holes" in the grid that need to be handled accordingly. Therefore, we keep the system at its full size (and its full grid), and we discuss the specific multigrid convergence issues in the next section.

Moreover, an efficient implementation of the multigrid methods requires that the sparse matrix data structure does

not change at run time, since otherwise the acceleration data structures that are used by the multigrid solver [GW10] have to be recomputed, too. This, for instance, is an issue when constructing an appropriate skeleton for the animation, since then the sets S_1 and S_2 change frequently due to the interactive repositioning and refinement of the skeleton (see Section 4.1).

Therefore, we propose to modify the system matrix K by zeroing the rows and columns that belong to vertices where displacements are enforced, i.e., the blocks K_{11} and K_{12}. This can be implemented efficiently using a row-compressed sparse matrix format. In order to keep the system of equations at full rank, the diagonal entries are set to a non-zero value α. The part of the right hand side f_1 is set to $f_1 = \alpha u_1$ to enforce the displacements u_1 in the solution process, resulting in the following system to be solved:

$$\begin{pmatrix} \alpha I & 0 \\ 0 & K_{22} \end{pmatrix} \begin{pmatrix} u_1 \\ u_2 \end{pmatrix} = \begin{pmatrix} \alpha u_1 \\ f_2 - K_{21} u_1 \end{pmatrix}.$$

Note that the system matrix still is symmetric, and thus we can use a Galerkin multigrid approach [TOS01]. Additionally, it allows for faster assembling of the system matrix, since only the upper diagonal part has to be recomputed in every time step to account for the element rotations.

In a second step we have to compute the internal forces that act on the constrained vertices, since these forces are used to update the skeleton as described later. The internal force vector f_1 is computed by utilizing a copy of the zeroed blocks $\hat{K}_{11}$ and $\hat{K}_{12}$ (the blocks in the system matrix K without the modifications of the time integration scheme):

$$f_1 = \hat{K}_{11} u_1 + \hat{K}_{12} u_2. \quad (1)$$

3.2. Boundary Conditions in the Multigrid Solver

We use a geometric multigrid solver on hexahedral or tetrahedral grids to efficiently solve the resulting system of linear equations. This requires a hierarchy of grids at different resolution levels, which is constructed as proposed by Georgii and Westermann [GW06], and which defines the restriction and interpolation operators. A simple approach to handle the displacement boundary conditions in the multigrid scheme is to replace the original equations of the boundary vertices with a dummy equation $\alpha I u = \alpha u_0$, where u_0 is the given displacement at the respective vertex, i.e. the boundary vertices are not removed from the equation system. This ensures that the interpolation and restriction matrices with respect to the unmodified coarse grids have full rank, and therefore a standard Galerkin coarsening leads to a convergent multigrid scheme [TOS01]. However, by using this strategy we observed decreased convergence compared to an unconstrained system.

Therefore, we propose a different approach, which is based on the elimination of boundary conditions, i.e., the boundary vertices (or constrained vertices) are removed as

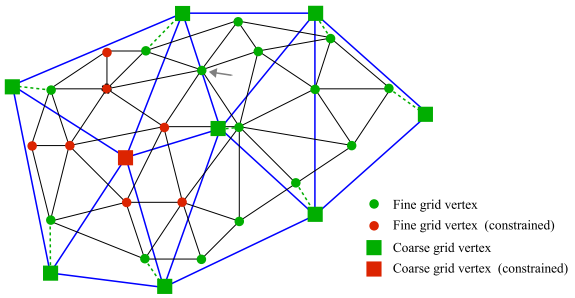

Figure 3: *Demonstration of the boundary condition handling for a two grid hierarchy. (For simplicity, a synthetic example in 2D is used.) The greedy approach searches for each coarse grid vertex a corresponding unconstrained fine grid vertex (green dotted line). If no such vertex is found, the coarse grid vertex is constrained (red). Note that in the example the fine grid vertex marked with the gray arrow cannot be considered since its interpolation weight (with respect to the constrained coarse grid vertex) is zero.*

degrees of freedom from the equation system. Note that it is then necessary to also remove vertices as degrees of freedom on the coarse grid levels of the multigrid hierarchy to obtain *full rank* interpolation operators. For the coarse grid levels, we determine the constrained vertices successively for each level up to the coarsest level by following a greedy strategy: At the beginning, all unconstrained vertices of the previous finer level are marked as "available", and the constrained vertices of the previous finer level are marked as "consumed". We then iterate over the vertices of the current level. For each vertex, we consider the set of vertices on the previous finer level, which interpolate from the currently considered vertex. If this set does not contain an "available" vertex, then the considered coarse grid vertex is marked as constrained, otherwise as unconstrained. In the latter case, the "available" vertex with the largest interpolation weight is marked as "consumed". This algorithm is illustrated in Figure 3. By eliminating the constrained vertices as proposed, we achieve an improved convergence rate of the multigrid solver compared to the first approach that adapts the equations of the constrained vertices.

3.2.1. Implementation

The goal of the following section is to describe how our novel approach for the elimination of boundary conditions can be efficiently integrated into existing multigrid simulation code. For performance reasons, we do not want to modify the sparse matrix data structures, which means that eliminated equation are still represented by a dummy equation. However, our novel approach ignores these equations in the multigrid scheme, i.e. they are not propagated to the coarser grids. For each constrained vertex we perform two steps. First, we modify the right-hand side of the vertices

in the 1-ring neighborhood to account for the displacement boundary condition (see Section 3.1). Second, we eliminate the equation at the constrained vertex by zeroing the respective row and column of the matrix (and keeping a diagonal entry).

In principle, we then want to ignore this eliminated equations on the coarser grids, too, which means that we have to adapt the restriction and interpolation operators in such a way that eliminated equations do not modify other coarse grid equations. That is, for the constrained vertices we zero the respective column in the restriction matrix and the respective row in the interpolation matrix.

However, then it might happen that as a result the restriction/interpolation matrices do not have full rank. Intuitively, one can see this fact as having more degrees of freedom on the coarser level than on the finer level, since degrees of freedoms on the finer level are restricted by the boundary conditions. In this situation, it is necessary to mark a reasonable number of coarse grid vertices as constrained, too, since otherwise the coarse grid operator becomes singular. We ensure this in a further step using our greedy approach as described above.

If we have marked a coarse grid vertex as constrained as result of the greedy approach, we have to ensure that this coarse grid vertex gets a dummy equation, too (since otherwise the coarse grid operator will be rank deficient). Therefore, we choose an arbitrary constrained fine grid vertex and manipulate the restriction/interpolation matrix such that the fine grid (dummy) equation is propagated to the constrained coarse grid vertex. In principle, this is the same as eliminating the coarse grid vertex.

The benefit of this procedure is that we neither have to modify the matrix data structure nor the multigrid code to manually ignore the constrained vertices in the solution process. However, the approach has to manipulate the stream acceleration data structures [GW10] that is used by the multigrid solver to efficiently determine the system equations on the coarser grids. These coarse grid matrices are built by means of sparse matrix products involving the interpolation and restriction operators, which are derived from the geometric grid hierarchy. To efficiently compute the sparse matrix products we use a stream-like layout of a data structure that allows one to perform these operations in a very cache-efficient way. To incorporate the novel boundary condition handling into the optimized multigrid method, we have to update the stream data structure, since it embeds respective entries of the interpolation and restriction operators for performance reasons. Fortunately, this is easily possible since the stream layout does not change (the matrix data structures do not change), and thus we can easily update the respective entries of the interpolation and restriction operators.

4. Skeleton-based Deformation

To enable the described two-way coupling between a skeleton and a finite element mesh, we now describe the skeleton construction and the binding of the constructed skeleton to the mesh. Ideally, one would assume the finite element mesh vertices to lie on the skeleton's bone segments to properly couple the skeleton simulation with the finite element method as suggested by Capell et al. [CGC*02a]. In this case, however, the finite element mesh has to be adapted whenever the skeleton changes, thus prohibiting an interactive modification of the skeleton. To overcome this limitation, we provide a method to bind a skeleton to a finite element model that does not need to have any vertices on the skeleton.

4.1. Skeleton Editor

Skeleton construction and modification is performed interactively via a skeleton editor illustrated in Figure 4. The editor provides the possibility to define different joint types, i.e. either a hinge joint or a ball joint. Furthermore, the user can add joint limits such as minimal and maximal rotation angle in case of hinge joints, or a cone that defines the possible movements in case of ball joints. These constraints are ensured by an inverse kinematic module that is used to determine the skeleton's pose.

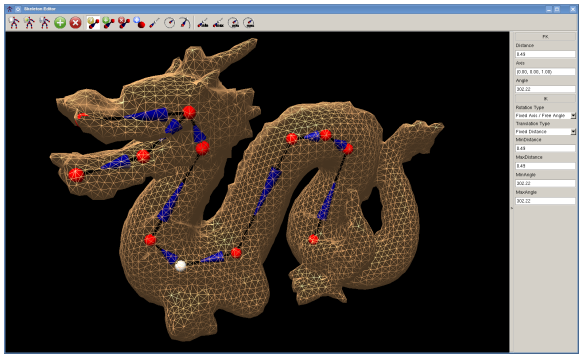

Figure 4: *A screenshot of the skeleton editor.*

4.2. Skeleton Binding

To bind the skeleton to the finite element simulation mesh, we apply the following simple approach: Firstly, we determine for every skeleton bone B_j the set of finite elements it intersects with. From these element set, we consider all element vertices v_i within a predefined distance to the bone B_j and project them onto the bone B_j to determine a linear interpolation weight

$$w_{ij} = \frac{(v_i - B_j^0)^T (B_j^1 - B_j^0)}{\|B_j^1 - B_j^0\|^2},$$

where B_j^0 and B_j^1 denote the position of the start and end joints of B_j, respectively. If $w_{ij} \notin [0,1]$, the vertex is not

bound to B_j. Otherwise, the weight w_{ij} is stored at the vertex together with the respective bone identifier. We will call these vertices the skeleton vertices in the following. In case a single vertex gets bound to several bones, we consider only the closest bone.

The linear interpolation weight is used to transfer movements from the skeleton to the finite element mesh. Skeleton vertices are handled as displacement boundary conditions in the simulation as described above, and the displacements for these vertices are determined by linearly interpolating the displacements of the respective joints using the weights w_{ij}. However, this simple approach has the drawback that it does not account for rotations around the bone axis, which can lead to artifacts in the simulation. To avoid this, one can either generate a mesh which is aligned with the skeleton, i.e. all skeleton vertices lie on the bone segments, or one can additionally rotate the skeleton vertices around the bone (in which case the rotation has to be included as degree of freedom in the skeleton simulation.)

Additionally, for each bone joint we accumulate the mass of the soft body that is relevant for this joint. For each vertex i of the finite element model, we first determine its mass by gathering the masses of the incident elements, and we then accumulate this mass to the best-fitting skeleton bone by choosing the closest bone j that satisfies $w_{ij} \in [0,1]$. Finally, the vertex mass is accumulated to the respective start and end joint of the bone j using the respective weights. These joint masses are used to move the skeleton based on forces acting on the skeleton as described in the next section. It is worth noting that the joint mass is only a rough approximation, i.e. not all finite element vertices propagate their mass since it might happen that there does not exist a joint with corresponding weight $w_{ij} \in [0,1]$. However, we observe that this approach effectively increases the stability of the proposed two-way coupling, which we describe in the next section.

4.3. Skeletal Constraints

In the Section 3.1 we have shown how to apply displacement boundary conditions to the skeleton vertices of the finite element mesh by fixating these vertices on the skeleton. To let the skeleton being moved by the soft body, we proceed in two steps: Firstly, the forces exerted at the skeleton vertices are determined using the finite element simulation (see Equation 1). Secondly, these forces are accumulated at the joints of the skeleton by considering the interpolation weights stored at each skeleton vertex. Finally, from the accumulated mass of the soft body at the skeleton's joints and the forces acting on it we obtain an update of the position of the skeleton's joint using an Verlet time integration scheme. Since we consider the mass of the deformable body to update the skeletal pose, we add inertia to the movement of the skeleton, which results in damping of our coupled simulations. To allow for an intuitive control of the amount of damping, we use a globally defined damping factor by which the forces are scaled.

In general, the new positions of the skeleton's joints could be used to define the new pose of the skeleton. However, since the skeleton itself is subject to a number of constraints (joint type, joint limits), we use inverse kinematics to recompute the skeleton's pose. Therefore, the goal position of every joint is set to the new position computed by the force integration. Then, we apply a Jacobian Transpose approach [Bus09] to compute the new skeleton's pose taking into account the joint limits. Since we only observe small movements of the skeleton from one time step to the next, this approach yields reasonable results. Upon convergence of the inverse kinematics computation the displacement boundary conditions of all skeleton vertices are updated with respect to the new skeleton pose. Then, the simulation starts over taking into account the new boundary conditions.

5. Results

We have tested our simulation framework on a standard desktop PC equipped with an Intel Core 2 Quad Q9450 2.66 GHz processor and 8 GB of RAM. In Table 1 we provide information on the models used and the respective performance achieved. The second column shows the number of tetrahedral elements that are used in the finite element simulation. The next column shows the degrees of freedom of the skeleton (computed by the inverse kinematic module). Then, we show the time it takes to initialize a skeleton simulation, i.e. to bound the skeleton to the finite element mesh as described in Section 4.2 and to update the system matrix as shown in Section 3.1. The last column contains the overall time to solve one time step of the dynamic corotated simulation (using two multigrid V(2,1)-cycles) including the time of the inverse kinematic module.

Model	#Elements	Skeleton DoFs	Init Skeleton	Solve
Manikin	14,062	10	1ms	49ms
Bar	24,576	4	2ms	56ms
Dragon	67,309	12	4ms	203ms

Table 1: *Performance Analysis. The bar example uses a nested grid hierarchy with 4 levels, while the other examples are based on non-nested grid hierarchies with 3 levels.*

We compare our multigrid solver to a conjugate gradient solver in Figure 5. We analyze the reduction of the relative error over time for both solvers using the bar example. Our multigrid solver (red) clearly outperforms a conjugate gradient solver with Jacobi preconditioner (blue). In Figure 6, we demonstrate the benefit of the novel approach to handle the boundary conditions at the coarser grids as described in Section 3.2. We analyze the reduction of the relative error of the solution achieved by an iteration of single multi-

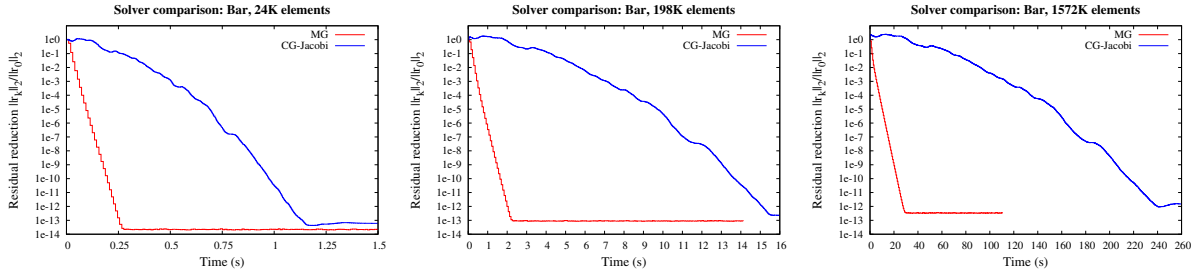

Figure 5: *Computational efficiency of different numerical solvers. Our multigrid approach (red) clearly outperforms a conjugate gradient solver with Jacobi preconditioner (blue).*

grid V(1,1)-cycles with one pre- and post-smoothing Gauss-Seidel step. It can be seen that our approach ("Eliminated Boundary Conditions") significantly improves the convergence rate of the multigrid solver compared to the simple approach ("Adapted Equations"). It is worth noting, however, that the latter one shows the same convergence rate as our approach if there are no constrained coarse grid vertices (since the interpolation with eliminated boundary conditions still has full rank). One of the main benefits of our approach to eliminate the boundary conditions is that we can improve the convergence rate without increasing the computational costs of a V-cycle.

We allow two different kinds of interaction modes in our implementation. Firstly, we can pick joints of the skeleton and move them to a specific position, which directly defines the goal positions for the inverse kinematic module. This mode can be used to directly control the skeletal pose. Secondly, we provide the proposed indirect deformation mode, where the user induces external forces on the object by mouse movements. The skeleton pose is then automatically recomputed as described in Section 4.3. Figure 7 shows some results achieved with our approach.

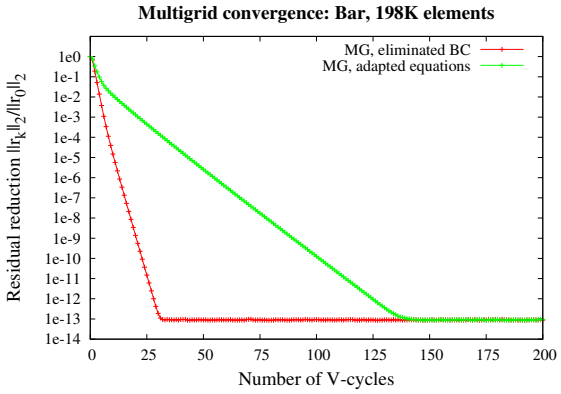

Figure 6: *Convergence analysis of the multigrid approach with boundary conditions (bar example). The new approach to eliminate the boundary conditions on the coarser grids (red) effectively improves the convergence rate compared to a standard Galerkin approach with adapted equations (green).*

6. Conclusion

We have demonstrated a novel approach to handle skeletal constraints in deformable object simulations. By using an inverse kinematic approach, we can update the pose of the skeleton, which then determines the boundary conditions in the finite element simulation. We provide a feedback mode, such that the internal forces acting on the skeleton can be used to automatically update the skeleton.

Moreover, we have shown how to efficiently embed the skeletal constraints into a geometric multigrid approach for the deformable object simulation. We proposed a novel approach to eliminate the constrained vertices while at the same time ensuring that the interpolation operator has full rank. This approach is a general means to handle boundary conditions in geometric multigrid techniques, for instance to fixate single vertices. By means of our approach, we can achieve interactive update rates for the simulation of large deformable bodies with skeletal constraints.

References

[ACP02] ALLEN B., CURLESS B., POPOVIĆ Z.: Articulated body deformation from range scan data. In *Proc. of SIGGRAPH '02* (2002), ACM, pp. 612–619.

[Bar96] BARAFF D.: Linear-time dynamics using lagrange multipliers. In *SIGGRAPH '96: Proceedings of the 23rd annual conference on Computer graphics and interactive techniques* (New York, NY, USA, 1996), ACM, pp. 137–146.

[Ben07] BENDER J.: Impulse-based dynamic simulation in linear time. *Journal of Computer Animation and Virtual Worlds 18*, 4-5 (2007), 225–233.

[BFA02] BRIDSON R., FEDKIW R., ANDERSON J.: Robust treatment of collisions, contact and friction for cloth animation. In *Proceedings of SIGGRAPH* (2002), pp. 594–603.

Figure 7: *A sequence of different skeletal constrained deformations of a manikin. We use a high-resolution render surface as proposed in [GW05].*

[BFGS03] BOLZ J., FARMER I., GRINSPUN E., SCHRÖODER P.: Sparse matrix solvers on the gpu: conjugate gradients and multigrid. In *Proceedings of SIGGRAPH* (2003), pp. 917–924.

[BNC96] BRO-NIELSEN M., COTIN S.: Real-time volumetric deformable models for surgery simulation using finite elements and condensation. In *Proceedings of Eurographics* (1996), pp. 57–66.

[Bra77] BRANDT A.: Multi-level adaptive solutions to boundary-value problems. *Mathematics of Computation 31*, 138 (1977), 333–390.

[Bus09] BUSS S. R.: *Introduction to inverse kinematics with jacobian transpose, pseudoinverse and damped least squares methods*. Tech. rep., University of California, 2009.

[CBC*05] CAPELL S., BURKHART M., CURLESS B., DUCHAMP T., POPOVIĆ Z.: Physically based rigging for deformable characters. In *Proceedings of the 2005 ACM SIGGRAPH/Eurographics symposium on Computer animation* (2005), ACM, pp. 301–310.

[CGC*02a] CAPELL S., GREEN S., CURLESS B., DUCHAMP T., POPOVIĆ Z.: Interactive skeleton-driven dynamic deformations. In *Proceedings of SIGGRAPH* (2002), pp. 586–593.

[CGC*02b] CAPELL S., GREEN S., CURLESS B., DUCHAMP T., POPOVIĆ Z.: A multiresolution framework for dynamic deformations. In *Proceedings of the ACM SIGGRAPH/Eurographics Symposium on Computer Animation* (2002), pp. 41–47.

[DDCB01] DEBUNNE G., DESBRUN M., CANI M.-P., BARR A. H.: Dynamic real-time deformations using space & time adaptive sampling. In *Proceedings of SIGGRAPH* (2001), pp. 31–36.

[DGBW08] DICK C., GEORGII J., BURGKART R., WESTERMANN R.: Computational steering for patient-specific implant planning in orthopedics. In *Proceedings of Visual Computing for Biomedicine 2008* (2008), pp. 83–92.

[FvdPT01] FALOUTSOS P., VAN DE PANNE M., TERZOPOULOS D.: Composable controllers for physics-based character animation. In *Proceedings of SIGGRAPH '01* (2001), pp. 251 – 260.

[GKS02] GRINSPUN E., KRYSL P., SCHRÖDER P.: CHARMS: a simple framework for adaptive simulation. In *Proceeding of SIGGRAPH* (2002), pp. 281–290.

[GOT*07] GALOPPO N., OTADUY M. A., TEKIN S., GROSS M. H., LIN M. C.: Soft articulated characters with fast contact handling. *Comput. Graph. Forum 26*, 3 (2007), 243–253.

[GTT89] GOURRET J.-P., THALMANN N. M., THALMANN D.: Simulation of object and human skin formations in a grasping task. *SIGGRAPH Comput. Graph. 23*, 3 (1989), 21–30.

[GW05] GEORGII J., WESTERMANN R.: Interactive Simulation and Rendering of Heterogeneous Deformable Bodies. In *Proceedings of Vision, Modeling and Visualization* (2005), pp. 383–390.

[GW06] GEORGII J., WESTERMANN R.: A Multigrid Framework for Real-Time Simulation of Deformable Bodies. *Computer & Graphics 30* (2006), 408–415.

[GW08] GEORGII J., WESTERMANN R.: Corotated finite elements made fast and stable. In *Proc. of the 5th Workshop On Virtual Reality Interaction and Physical Simulation* (2008), pp. 11–19.

[GW10] GEORGII J., WESTERMANN R.: A streaming approach for sparse matrix products and its application in Galerkin multigrid methods. *Electronic Transactions on Numerical Analysis, to appear* (2010).

[HS04] HAUTH M., STRASSER W.: Corotational simulation of deformable solids. In *Proceedings of WSCG* (2004), pp. 137–145.

[HWBO95] HODGINS J. K., WOOTEN W. L., BROGAN D. C., O'BRIEN J. F.: Animating human athletics. In *SIGGRAPH '95: Proceedings of the 22nd annual conference on Computer graphics and interactive techniques* (New York, NY, USA, 1995), ACM, pp. 71–78.

[IC87] ISAACS P. M., COHEN M. F.: Controlling dynamic simulation with kinematic constraints. *SIGGRAPH Comput. Graph. 21*, 4 (1987), 215–224.

[JP99] JAMES D. L., PAI D. K.: ArtDefo: accurate real time deformable objects. In *Proceedings of SIGGRAPH* (1999), pp. 65–72.

[JZvdP*08] JU T., ZHOU Q.-Y., VAN DE PANNE M., COHEN-OR D., NEUMANN U.: Reusable skinning templates using cage-based deformations. In *Proc. of SIGGRAPH Asia '08* (2008), ACM, pp. 1–10.

[KH08] KAZHDAN M., HOPPE H.: Streaming multigrid for gradient-domain operations on large images. *ACM Trans. Graph. 27*, 3 (2008), 1–10.

[MDM*02] MÜLLER M., DORSEY J., MCMILLAN L., JAGNOW R., CUTLER B.: Stable real-time deformations. In *Proceedings of ACM SIGGRAPH/Eurographics Symposium on Computer Animation* (2002), pp. 49–54.

[MG04] MÜLLER M., GROSS M.: Interactive virtual materials. In *Proceedings of Graphics Interface* (2004), pp. 239–246.

[MHTG05] MÜLLER M., HEIDELBERGER B., TESCHNER M., GROSS M.: Meshless deformations based on shape matching. *ACM Trans. Graph. 24*, 3 (2005), 471–478.

[MKN*04] MÜLLER M., KEISER R., NEALEN A., PAULY M., GROSS M., ALEXA M.: Point based animation of elastic, plastic and melting objects. In *Proceedings of the ACM SIGGRAPH/Eurographics Symposium on Computer Animation* (2004), pp. 141–151.

[Mül08] MÜLLER M.: Hierarchical position based dynamics. In *Proc. of the 5th Workshop On Virtual Reality Interaction and Physical Simulation* (2008).

[NKJF09] NESME M., KRY P. G., JEŘÁBKOVÁ L., FAURE F.: Preserving topology and elasticity for embedded deformable models. In *SIGGRAPH '09: ACM SIGGRAPH 2009 papers* (New York, NY, USA, 2009), ACM, pp. 1–9.

[NMK*05] NEALEN A., MÜLLER M., KEISER R., BOXERMAN E., CARLSON M.: Physically based deformable models in computer graphics. In *Proceedings of Eurographics* (2005), pp. 71–94.

[SF98] SINGH K., FIUME E.: Wires: a geometric deformation technique. In *Proc. of SIGGRAPH '98* (1998), ACM, pp. 405–414.

[SPCM97] SCHEEPERS F., PARENT R. E., CARLSON W. E., MAY S. F.: Anatomy-based modeling of the human musculature. In *Proc. of SIGGRAPH '97* (1997), ACM, pp. 163–172.

[SSF08] SHINAR T., SCHROEDER C., FEDKIW R.: Two-way coupling of rigid and deformable bodies. In *Proc. of the 2008 ACM SIGGRAPH/Eurographics Symposium on Computer Animation* (2008), Eurographics Association, pp. 95–103.

[SYBF06] SHI L., YU Y., BELL N., FENG W.-W.: A fast multigrid algorithm for mesh deformation. *ACM Trans. Graph. 25*, 3 (2006), 1108–1117.

[SZHB06] SONG C., ZHANG H.-X., HUANG J., BAO H.-J.: Meshless simulation for skeleton driven elastic deformation. *Journal of Zhejiang University - Science A 7* (2006), 1596–1602.

[TF88] TERZOPOULOS D., FLEISCHER K.: Modeling inelastic deformation: Viscoelasticity, plasticity, fracture. In *Proceedings of SIGGRAPH* (1988), pp. 269–278.

[TOS01] TROTTENBERG U., OOSTERLEE C., SCHÜLLER A.: *Multigrid*. Academic Press, 2001.

[TPBF87] TERZOPOULOS D., PLATT J., BARR A., FLEISCHER K.: Elastically deformable models. In *Proceedings of SIGGRAPH* (1987), pp. 205–214.

[WDGT01] WU X., DOWNES M. S., GOKTEKIN T., TENDICK F.: Adaptive nonlinear finite elements for deformable body simulation using dynamic progressive meshes. In *Proceedings of Eurographics* (2001), pp. 349–358.

[WGF08] WEINSTEIN R., GUENDELMAN E., FEDKIW R.: Impulse-based control of joints and muscles. *IEEE Transactions on Visualization and Computer Graphics 14*, 1 (2008), 37–46.

[WT04] WU X., TENDICK F.: Multigrid integration for interactive deformable body simulation. In *Proceedings of International Symposium on Medical Simulation* (2004), pp. 92–104.

[WTF06] WEINSTEIN R., TERAN J., FEDKIW R.: Dynamic simulation of articulated rigid bodies with contact and collision. *IEEE Transactions on Visualization and Computer Graphics 12*, 3 (2006), 365–374.

[WVG97] WILHELMS J., VAN GELDER A.: Anatomically based modeling. In *Proc. of SIGGRAPH '97* (1997), ACM, pp. 173–180.

[ZSTB10] ZHU Y., SIFAKIS E., TERAN J., BRANDT A.: An efficient multigrid method for the simulation of high-resolution elastic solids. *ACM Trans. Graph. 29*, 2 (2010), 1–18.