Hyper-Reduced Projective Dynamics

CHRISTOPHER BRANDT, ELMAR EISEMANN, and KLAUS HILDEBRANDT,
Delft University of Technology, The Netherlands

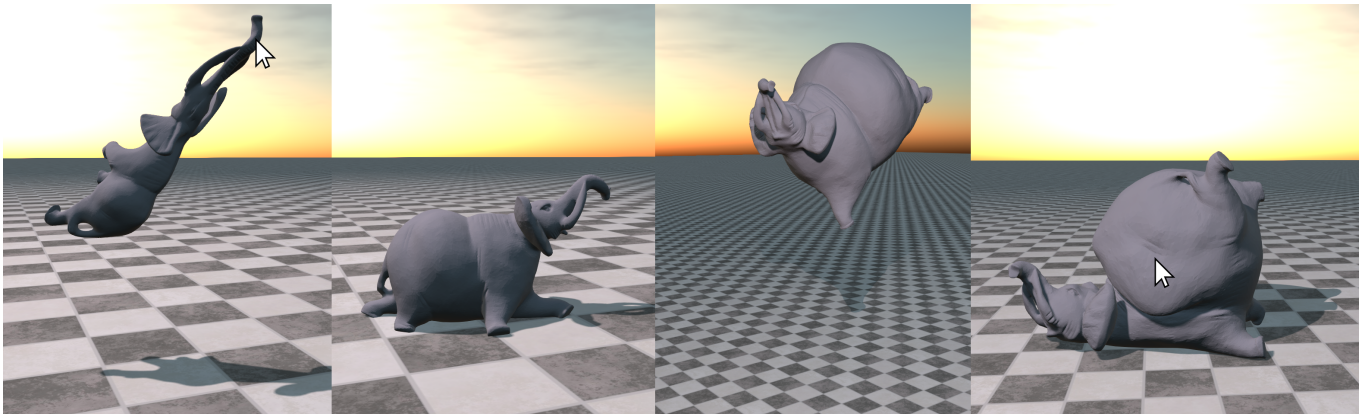

Fig. 1. Our Hyper-Reduced Projective Dynamics method is able to handle different types of constraints simultaneously (volume preservation for tetrahedrons and strain resistance for boundary triangles) for complex geometries (52k vertices) in real-time. During the simulation we change the target volume of the tetrahedrons, which causes a balloon-like blowup effect. We resolve ground collisions and allow user interaction by click and drag. The simulation runs at 62 fps including rendering. The total precomputation time of our method is 21 seconds.

We present a method for the real-time simulation of deformable objects that combines the robustness, generality, and high performance of Projective Dynamics with the efficiency and scalability offered by model reduction techniques. The method decouples the cost for time integration from the mesh resolution and can simulate large meshes in real-time. The proposed hyper-reduction of Projective Dynamics combines a novel fast approximation method for constraint projections and a scalable construction of sparse subspace bases. The resulting system achieves real-time rates for large subspaces enabling rich dynamics and can resolve general user interactions, collision constraints, external forces and changes to the materials. The construction of the hyper-reduced system does not require user-interaction and refrains from using training data or modal analysis, which results in a fast preprocessing stage.

CCS Concepts: • **Computing methodologies** → **Real-time simulation**; **Physical simulation**;

Additional Key Words and Phrases: Real-time simulation, projective dynamics, model reduction, subspace dynamics, reduced-order model

ACM Reference Format:
Christopher Brandt, Elmar Eisemann, and Klaus Hildebrandt. 2018. Hyper-Reduced Projective Dynamics. *ACM Trans. Graph.* 37, 4, Article 80 (August 2018), 13 pages. https://doi.org/10.1145/3197517.3201387

Authors' address: Christopher Brandt; Elmar Eisemann; Klaus Hildebrandt, Delft University of Technology, Department of Intelligent Systems, Van Mourik Broekmanweg 6, Delft, 2628 XE, The Netherlands, {c.brandt,e.eisemann,k.a.hildebrandt}@tudelft.nl.

1 INTRODUCTION

The simulation of deformable objects is an important factor in various entertainment and training applications of computer graphics. Of particular interest are real-time simulations since these allow the user to interact with the simulation and thereby greatly enrich the experience in games, virtual reality, artistic applications and medical training. The combination of the nonlinear nature of deformable objects and the geometric complexity of objects in these scenarios make real-time simulation a challenging problem. In addition to efficiency, robustness is important for real-time simulation since interactivity leads to unforeseeable states of the system.

Projective Dynamics, introduced in [Bouaziz et al. 2014; Liu et al. 2013], is a simulation framework that is *general*, since it allows for the simulation of different types of deformable objects (solids, shells and rods) and materials. It is *robust*, as it capable of adequately handling large time steps, large deformations and degenerate geometries, and it is *fast*. On the technical level, a variational implicit time integration scheme is used and the resulting optimization problem is solved by an alternating local/global approach. The local steps in the optimization can be executed in parallel, and for *constraint-based* potentials, the system matrix of the linear system to be solved in the global steps is constant, allowing for the re-use of a sparse factorization of the matrix. While this makes Projective Dynamics highly efficient for real-time simulation, it is still limited to objects that can be represented by coarse geometric models.

Model reduction is concerned with the design of fast approximation algorithms for simulation, control, and design problems. Reduced-order models divide the computation into an offline and an online stage. In the offline stage, a reduced system is constructed that approximates the original system and is fast to evaluate. In the

online stage, only the reduced system is evaluated. Reduced-order models can drastically reduce the computational cost while approximating the original system. This makes model reduction attractive for real-time applications.

We propose a hyper-reduction of Projective Dynamics that enables the real-time simulation of deformable objects with complex geometry. The scheme is designed such that the computational cost for time integration is independent of the complexity of the mesh representing the object. It proceeds in two stages. First, a subspace to which the simulation is restricted is constructed. While this step reduces the degrees of freedom of the simulation, it does not necessarily lower the cost for integrating the system since the complexity for evaluating the constraint projections, which describe the acting forces, is not reduced. Therefore, a second level of approximation, a *hyper-reduction*, is needed. We introduce a novel approach, the *constraint projections fitting method*, for this second stage. The idea is to construct a second subspace during the preprocessing stage—not for vertex positions—but for the constraint projections. Then, in the online stage, only a limited number of constraints are evaluated and a fitting problem in the second subspace is solved to obtain approximations of the constraint projections. This scheme is inspired by the empirical interpolation method from applied mathematics and continuum mechanics [Barrault et al. 2004; Chaturantabut and Sorensen 2010] (though our scheme is neither empirical nor interpolating) and is the first application of this type of method for reduced simulation in graphics.

To implement the hyper-reduced Projective Dynamics framework, constructions for the two subspaces are needed. We propose two subspace constructions that refrain from using prior knowledge about the nature of the simulation (types of user interaction, external forces, collision constraints, etc.). Only the mesh and the types of materials to be simulated are known. While this setting does not allow our method to profit from specialization to a specific scenario, the resulting benefits are an automatic and accelerated preprocessing stage, which avoids costly probing of the simulation or modal analysis.

Our position subspace construction follows constructions of skinning spaces [Jacobson et al. 2011; Wang et al. 2015]. However, instead of solving a systems for each basis function, we use compactly supported radial basis functions for the weights. As a result, subspaces of large scale (up to the original size of the mesh) can be constructed efficiently and the resulting basis vectors are sparse. We generalize the construction to obtain subspaces of constraint projections. Since the subspace constructions are based on sampling, our subspace constructions make the assumption that the deformations vary smoothly along the object. Note, however, that this is not a general limitation of the proposed hyper-reduction. For example, subspaces constructed from snapshots (taken from a full simulation) could add high-frequency deformations, like sharp bends, to the subspaces. The general assumption for the hyper-reduced simulation is that the object remains close to a low-dimensional submanifold in configuration space.

The resulting hyper-reduced system is highly efficient: In our experiments, we achieve real-time rates of 60 fps for simulations in 4k-dimensional subspaces, which is an order of magnitude higher than what is reported for recent hyper-reduced schemes like [von Tycowicz et al. 2013; Wu et al. 2015]. Precomputation time for a mesh with 19k vertices is 6.8 seconds and one minute for a complex geometry with 200k vertices. The examples shown in the supplementary video demonstrate that our hyper-reduced system enables rich dynamics and can plausibly resolve unexpected events, such as collisions, drastic user interactions and online changes of the material stiffness and the geometry (*e.g.* the volume).

2 RELATED WORK

Our method combines the benefits of Position-Based and Projective Dynamics methods with the efficiency offered by model reduction techniques. In the following, we provide an brief overview of literature in these fields and embed our method into prior work.

Position-Based Dynamics. Position-Based Dynamics [Müller et al. 2007; Stam 2009] were introduced as general and efficient methods to enable the simulation of deformable objects in real-time. They can be seen as extensions of Shape Matching Methods [Müller et al. 2005; Rivers and James 2007]. Position-Based Dynamics omits the velocity layer and instead works directly on the positions. Inner forces are expressed as equality or inequality constraints, which are being enforced in a Gauss-Seidel-type fashion. This *constraint projection* step can be carried out in parallel and heavily sped up via GPU implementations. Since then the method has been extended in terms of robustness, convergence and generality [Kim et al. 2012; Müller 2008; Müller and Chentanez 2011]. Most recently the problem of controlling stiffness parameters independently of the iteration count has been addressed [Macklin et al. 2016]. The method has been extended to fluids [Macklin and Müller 2013] and continuum based materials [Bender et al. 2014; Müller et al. 2014]. For a recent survey on Position-Based simulation techniques we refer to [Bender et al. 2017].

Projective Dynamics. Projective Dynamics [Bouaziz et al. 2014; Liu et al. 2013] is a method for implicit time integration of physical systems, which combines a variational integration scheme [Martin et al. 2011] with a specific class of potentials modeling the inner forces, resulting in a highly flexible and fast technique that is capable of simulating strings, cloth, elastic deformable objects and recently fluids [Weiler et al. 2016]. Narain et al. [2016] and Overby et al. [2017] apply the Alternating Direction Method of Multipliers (*ADMM*) optimization algorithm to the variational integration scheme and show that the resulting method closely relates to Projective Dynamics. From this point of view, it can be extended to nonlinear constitutive materials and hard constraints. Liu et al. [2017] show that Projective Dynamics can be seen as a quasi-Newton method for implicit variational time integration for certain types of potentials. This also allows for a generalization of the method to handle a large class of materials, such as Neo-Hookean or spline-based materials. While both generalizations give rise to similar reduced methods for nonlinear materials, our method is a hyper-reduction for the original Projective Dynamics method, and we make use of specific properties of the system to gain the efficiency shown in our results. Wang [2015] presents a GPU implementation of the Projective Dynamics method, where Chebychev iterations are combined with Jacobi or Gauss-Seidel iterations to approximate the system in the

global step. This enhances convergence and reduces computation times for the Projective Dynamics time steps. It allows, for example, the simulation of a complex 20k vertex mesh at 38 fps. However, being an unreduced method, the computation time for each iteration still depends on the mesh resolution, such that real-time frame rates are only possible up to a certain number of vertices. In contrast, our Hyper-Reduced Projective Dynamics method achieves computation times that are independent of the mesh resolution. For example, we are able to simulate a mesh with 200k vertices in 60 fps, including collision handling and rendering, using a CPU implementation on a consumer desktop computer.

Model reduction for deformables. An early application of model reduction to the simulation of deformable objects in graphics is [Pentland and Williams 1989], in which a linearized simulation is restricted to a subspace spanned by the low-frequency linear vibration modes. Since the linear modes do not capture non-linearities well, techniques for augmenting linear modal bases for nonlinear deformable object simulation have been developed: Choi and Ko [2005] proposed *modal warping*, Barbič and James [2005] basis augmentation with *modal derivatives*, Huang et al. [2011] and Pan et al. [2015] *rotation strain coordinates*, Tycowicz et al. [2013] basis enrichment via linear space transformations, and Yang et al. [2015; 2013] a *linear inertia mode* technique. Since vibration modes are globally supported, modal bases consist of dense vectors. Brandt and Hildebrandt [2017] introduced a scheme for the compression of vibration modes resulting in sparse basis vectors. An alternative to using modal analysis are *empirical eigenmodes* [Barbič and James 2005; Krysl et al. 2001], which collect snapshots of the systems to be reduced and use principal component analysis for the construction of a reduced basis. Neumann et al. [2013] proposed a sparse PCA approach to get sparse empirical deformation bases. A third type of subspaces are skinning spaces. Skinning spaces constructed by sampling the geometry were used for fast simulation, for example, by Gilles et al. [2011] and Jacobson et al. [2012].

Hyper-reduced systems combine dimensional reduction with a scheme for fast reduced force approximation. For St. Venant–Kirchhoff materials, the elastic potential is a quartic polynomial in the vertex displacements. Barbič and James [2005] suggest the precomputation of the coefficients of this polynomial in the reduced space, which enables exact evaluation of potential, forces, and the Hessian at a cost depending only on the dimension of the subspace. For general materials, one needs to resort to approximation. The optimized cubature, introduced by An et al. [2008], approximates the subspace forces by a linear combination of projections of a selection of local force vectors. The weights and sample locations of local force vectors are determined by cubature training, in which the approximation error against a set of snapshots of force vectors is minimized. Tycowicz et al. [2013] introduced a *non-negative hard thresholding pursuit* solver that significantly speeds-up cubature training. Yang et al. [2015] further reduce the training time by an effective strategy for training data generation, which generates only the data required to reach a given error margin. Kim and Delaney [2013] and Harmon and Zorin [2013] use importance-based sampling to determine cubature point locations and optimize only the weights. Wu et al. [2015] propose a domain decomposition approach to improve performance of cubature training and evaluation. Since for our hyper-reduction of Projective Dynamics, approximations of constraint projections as opposed to force vectors are needed, polynomial reduction and cubature cannot be applied. Conversely, the proposed fitting method holds potential for other model reduction problems in graphics.

In addition to simulation, model reduction has been used for controlling and editing the motion of deformable objects [Barbič et al. 2012; Hildebrandt et al. 2012; Li et al. 2014; Schulz et al. 2014], interactive material design [Xu et al. 2015], sound synthesis [Chadwick et al. 2009], clothing simulation [Hahn et al. 2014], deformation-based shape modeling [Hildebrandt et al. 2011; Huang et al. 2006; Jacobson et al. 2012; Wang et al. 2015], elasticity-based shape interpolation [von Radziewsky et al. 2016; von Tycowicz et al. 2015], curve processing in shape space [Brandt et al. 2016], and shape optimization [Chen et al. 2017; Musialski et al. 2015].

Coarse simulation. An alternative to the methods considered in this work is to simulate a coarse mesh and couple the coarse mesh to the high-resolution geometry such that deformations of the coarse mesh can be propagated to deformations of the fine mesh [Capell et al. 2002; Debunne et al. 2001; Kharevych et al. 2009; Nesme et al. 2009; Rémillard and Kry 2013; Wojtan and Turk 2008]. To put these approaches in context with our model reduction scheme, one can observe that the coupling of the coarse and fine mesh results in a specific subspace for the fine mesh, where the reduced coordinates are the vertex positions of the coarse mesh. The simulations in the subspaces, however, differ significantly. Instead of computing a coarse simulation, our hyper-reduced scheme aims at approximating the dynamics of the high-resolution mesh. In a supplementary document and video, we present a comparison of our method and a coarse simulation.

3 BACKGROUND: PROJECTIVE DYNAMICS

For a mesh with time-dependent vertex positions $\mathbf{q} \in \mathbb{R}^{n\times 3}$, the goal is to find trajectories of the laws of motion

$$\mathbf{M}\ddot{\mathbf{q}} = \mathbf{f}_{\text{int}}(\mathbf{q}) + \mathbf{f}_{\text{ext}} \tag{1}$$

with initial conditions $\mathbf{q}(0) = \mathbf{q}_0$ and $\dot{\mathbf{q}}(0) = \mathbf{v}_0$. Here $\mathbf{f}_{\text{ext}}$ are (possibly time-dependent) external forces, such as gravity, and $\mathbf{f}_{\text{int}}(\mathbf{q}) = -\sum_i \nabla W_i(\mathbf{q})$ are internal forces acting on the mesh, such as elastic forces acting on deformable objects. Equation (1) can be solved via implicit Euler integration in a variational manner by solving the following optimization problem

$$\mathbf{q}(t+h) = \arg\min_{\mathbf{q}} \frac{1}{2h^2} \left\| \mathbf{M}^{\frac{1}{2}} (\mathbf{q} - \mathbf{s}) \right\|_F^2 + \sum_i W_i(\mathbf{q}) \tag{2}$$

$$\mathbf{v}(t+h) = \frac{1}{h}(\mathbf{q}(t+h) - \mathbf{q}(t)) \tag{3}$$

where $\mathbf{s} = \mathbf{q}(t) + h\mathbf{v}(t) + h^2\mathbf{M}^{-1}\mathbf{f}_{\text{ext}}$. The functional that is optimized in Equation (2) can be read as a trade-off between the momentum potential, which expresses that the object should follow its current trajectory (possibly altered by external forces), and the elastic potential, which maintains the shape of the object.

Projective Dynamics is particularly efficient for potentials W_i that have the following form:

$$W_i(\mathbf{q}) = \sum_j \frac{\lambda_j}{2} \left\| \mathbf{S}_j \mathbf{q} - \mathbf{p_j}(\mathbf{q}) \right\|^2 \tag{4}$$

$$\text{where } \mathbf{p}_j(\mathbf{q}) = \arg\min_{\mathbf{p}\in\Omega} \left\| \mathbf{S}_j \mathbf{q} - \mathbf{p} \right\|^2 \tag{5}$$

where the triplets $\{\lambda_j, \mathbf{S}_j, \mathbf{p_j}\}$ are called *constraints* and

- λ_j is a scalar weight denoting the stiffness of the constraint (and contains the area or volume associated to the constraint).
- $\mathbf{S}_j$ is a $p \times n$ matrix, which usually is a linear discrete differential operator computing deformation gradients, Laplacians, or other quantities for a certain element of the mesh.
- $\mathbf{p}_j$, the *constraint projection*, is a, typically nonlinear, function from $\mathbb{R}^{n\times 3}$ to $\mathbb{R}^{p\times 3}$ that projects the differential property $\mathbf{S}_j\mathbf{q}$ onto a constraint manifold Ω.

With these types of potentials, one can model a variety of materials, *e.g.* they can be used as bending and strain energies for thin shells, elastic energies for tetrahedral meshes, spring energies, and unconventional energies such as example-based materials [Martin et al. 2011]. The derivation and implementation of specific constraints can be found in the original Projective Dynamics paper [Bouaziz et al. 2014]. Most importantly, these constraints admit a robust minimization scheme for solving the optimization problem stated in Equation (2), which is a local/global approach. In the local step, the constraint projections can be evaluated independently in parallel, while in the global step, a system is solved that minimizes the objective for fixed constraints. This system decouples into the spatial dimensions x, y, and z, such that it can be solved by three linear solves of dimension n in parallel. The resulting method is listed in Algorithm 1.

Algorithm 1 Unreduced Projective Dynamics

Input: Current vertex positions $\mathbf{q}(t)$ and current velocities $\mathbf{v}(t)$
Let $\mathbf{s} = \mathbf{q}(t) + h\mathbf{v}(t) + h^2\mathbf{M}^{-1}\mathbf{f}_{\text{ext}}$
Let $\hat{\mathbf{q}} = \mathbf{s}$
for i = 1 **to** *numIterations* **do**
  Local step: Evaluate all constraint projections $\mathbf{p}_i = \mathbf{p}_i(\hat{\mathbf{q}})$
  Global step: Solve
  $\left(\frac{\mathbf{M}}{h^2} + \sum_i \lambda_i \mathbf{S}_i^T \mathbf{S}_i\right) \mathbf{q} = \frac{\mathbf{M}}{h^2}\mathbf{s} + \sum_i \lambda_i \mathbf{S}_i^T \mathbf{p}_i$
  Let $\hat{\mathbf{q}} = \mathbf{q}$
end for
Output: Updated vertex positions $\mathbf{q}(t+h) = \hat{\mathbf{q}}$ and velocities $\mathbf{v}(t+h) = (\mathbf{q}(t+h) - \mathbf{q}(t))/h$

4 VERTEX POSITIONS SUBSPACE CONSTRUCTION

In order to reduce the degrees of freedom in the optimization problem stated in Equation (2), we opt for a linear subspace $\mathbf{U} \in \mathbb{R}^{n\times 4k}$, such that vertex positions $\mathbf{q} \in \mathbb{R}^{n\times 3}$ can be approximated as $\mathbf{q} \approx \mathbf{U}\tilde{\mathbf{q}}$, where $\tilde{\mathbf{q}} \in \mathbb{R}^{4k\times 3}$ and $4k \ll n$. Such linear subspaces have been employed for many problems in computer graphics and specifically for the reduction of simulation and modeling applications, see Section 2.

Subspace construction. We opt for creating a subspace from skinning weights (see Appendix B), similar to the subspaces used in [Jacobson et al. 2011; Wang et al. 2015]. However, we define the required weights in a way that produces compactly supported basis functions, is well suited for scaling the trade-off between accuracy and performance, requires no user input, and can be computed in a few seconds at most. On a conceptual level we want to introduce degrees of freedom from affine transformations acting on equidistantly distributed handles (or areas) on the mesh, with smoothly varying, localized influence on the nearby vertices.

To this end, we first choose k sample vertices $\mathbf{s}_j$ which are distributed approximately equidistant over the mesh, using furthest point sampling. That is, the first vertex sample is chosen randomly and then we iteratively add the vertex which has furthest distance from all previously chosen samples on the mesh as the next sample. To ensure that the sampling is fair, even in presence of complex details, it is important that distances are measured as the lengths of geodesics on (or, in case of volumetric meshes, through) the mesh. To quickly evaluate geodesic distances to all of the sample points, we use the Short Term Vector Dijkstra (STVD) method proposed by Campen et al. [2013], which is a modification of the original Dijkstra algorithm that replaces the distance update with a method that uses a stack of predecessor edges to compute good approximations of the actual geodesic distances.

Preliminary weights for each vertex i and each of the k samples are then acquired from radial basis functions around the handles. Specifically they are defined as $\tilde{\mathbf{w}}_i = \left(B_{\mathbf{s}_1,r}(\mathbf{q}_i), ..., B_{\mathbf{s}_k,r}(\mathbf{q}_i)\right)$, where r is a radius whose choice we detail below and $B_{\mathbf{y},r}(\mathbf{x}) = b_r(d(\mathbf{x},\mathbf{y}))$ is a scalar function on the mesh vertices. On $[0,r]$, we choose b_r as the unique cubic polynomial with $b_r(r) = b_r'(0) = b_r'(r) = 0$ and $b_r(0) = 1$ and define $b_r(t)$ to be 0 for $t > r$. Other choices are possible, as long as they are smooth, monotone and interpolate between 1 and 0 on $[0,r]$. The final weights $\mathbf{w}_i$ are then normalized as $\mathbf{w}_i = \tilde{\mathbf{w}}_i \cdot (1./\sum_j \tilde{w}_i^j)$ in order to ensure reproducibility of the rest shape. To be able to evaluate the functions $B_{\mathbf{s}_j,r}$ we need to evaluate the geodesic distances $d(\mathbf{s}_j, \mathbf{q}_i)$ for all vertex positions $\mathbf{q}_i$. For this, we also employ the STVD method, where the search can be terminated early, since vertices with distance greater than r do not need to enter the queue (at the end unvisited vertices receive the value $B_{\mathbf{s}_j,r}(\mathbf{q}_i) = 0$). This setup enables the computation of smooth and locally supported weight functions, while not relying on solving box-constraint quadratic problems [Jacobson et al. 2011] or linear systems [Wang et al. 2015] on the mesh for every basis function, which becomes costly for high resolution meshes.

The choice of the parameter r is crucial for two reasons: on the one hand, the sparsity of the subspace matrix, which depends on this parameter, influences the performance of the Hyper-Reduced Projective Dynamics algorithm, since the cost of updating sampled vertex positions is tied to the sparsity pattern. On the other hand, r should be chosen large enough to ensure that each vertex is within the support of at least one weight function. In practice we found that $r = 2 \cdot \max_i \min_j d(\mathbf{s}_j, \mathbf{q}_i)$, offers a good compromise between sparsity of the base matrix and coverage of the vertices by a sufficient number of non-zero weights. Note that the quantities $\min_j d(\mathbf{s}_j, \mathbf{q}_i)$

can be evaluated cheaply by performing a last update to the available distances, adding the last source s_k.

After the weights have been computed, we construct the subspace basis matrix $\mathbf{U}$ such that it contains the degrees of freedom from the skinning transformations according to these weights. This construction is detailed in Appendix B. The result is a sparse matrix of size $n \times (4 \cdot k)$ which offers $12 \cdot k$ degrees of freedom for the simulated mesh, (k is the number of sample points chosen in the construction of the weight functions above). Restricting the system to the degrees of freedom offered by the subspace matrix $\mathbf{U}$, the global step of the Projective Dynamics method is replaced by

$$\mathbf{U}^T \left(\frac{\mathbf{M}}{h^2} + \sum_i \lambda_i \mathbf{S}_i^T \mathbf{S}_i \right) \mathbf{U}\tilde{\mathbf{q}} = \mathbf{U}^T \frac{\mathbf{M}}{h^2} \mathbf{U}\tilde{\mathbf{s}} + \mathbf{U}^T \sum_i \lambda_i \mathbf{S}_i^T \mathbf{p}_i, \quad (6)$$

where $\tilde{\mathbf{u}}$ and $\tilde{\mathbf{s}}$ are the current subspace coordinates for positions and the momentum term $\mathbf{s}$.

In Section 7, we investigate how well we can approximate states $\mathbf{q}$ of a full simulation by their projections $\tilde{\mathbf{q}}$ into subspaces obtained from our construction, using different values of k, see Table 2.

5 CONSTRAINT PROJECTIONS FITTING METHOD

In the previous section we constructed a subspace for the vertex positions in order to reduce the dimensionality of the problem. This reduces the cost for the linear solve in the global step and leads to faster convergence to a subspace optimum. However, the cost of evaluating the constraint projections in the local step remains unchanged. In particular, it still depends on the resolution of the mesh instead of the desired detail of the dynamics.

Overview. We propose a novel way of directly approximating the term $\mathbf{U}^T \sum_i \lambda_i \mathbf{S}_i^T \mathbf{p}_i$ on the right hand side of the reduced system (6) for the global step. The approach consists of a precomputation step and several steps in the online phase.

- **Precomputation:** We construct a subspace for constraint projections $\mathbf{V}$ and select s constraint samples to evaluate.
- **Online Phase:**
  1. Evaluate the positions of all vertices that appear in any of the sampled constraints.
  2. Evaluate the sampled constraint projections.
  3. Solve a fitting problem to find a best approximating vector $\tilde{\mathbf{p}}$ in the space spanned by $\mathbf{V}$.
  4. Evaluate $\mathbf{r} := \mathbf{U}^T \mathbf{S}^T \mathbf{V}\tilde{\mathbf{p}}$. ($\mathbf{S}$ will be defined in (7) below.)

Also note that both the precomputation and the online steps are separated for each *type* of constraint. For example, if both bending and strain constraints for a triangular mesh are present, different subspaces $\mathbf{V}$ are constructed, separate fitting problems are solved and their contributions to the right hand side are evaluated individually and then summed.

Subspace for constraint projections. When approximating the term $\mathbf{U}^T \sum_i \lambda_i \mathbf{S}_i^T \mathbf{p}_i$, one would like to skip the costly evaluation of *all* $\mathbf{p}_i(\mathbf{q})$, as well as prevent the large vector matrix multiplication with $\mathbf{U}^T$. To address the first goal, we design a basis $\mathbf{V}$ that spans a subspace of constraint projections, such that $(\mathbf{p}_0(\mathbf{q}), \mathbf{p}_1(\mathbf{q}), \ldots) \approx \mathbf{V}\tilde{\mathbf{p}}$. The space needs to be general enough that it contains good approximations of any such vectors that are encountered during a simulation, but concise enough that solving a least-squares fitting problem using a few constraint projections into this subspace yields these approximations. $\mathbf{V}$ can be constructed by using snapshots of forces (or, in our case, constraint projections) from full simulations (see *e.g.* the Discrete Empirical Interpolation Method [Chaturantabut and Sorensen 2010]). Specifically for high-resolution meshes, conducting full simulations in which enough interesting states can be observed to construct a sufficiently rich space is a complicated and time-consuming endeavor. Additionally, it requires knowledge about the type of interaction, collision constraints and external forces that will be encountered during the reduced simulation, which we do not want to assume in order to stay as general as possible. Note that in the reduced dynamics, the contribution of the constraint projections to the right hand side will always be filtered by $\mathbf{U}^T$. This suggests that one might want to use the columns of $\mathbf{U}$ directly to obtain vectors of constraint projections from these. However, this can only yield a meaningful subspace if the columns of $\mathbf{U}$ directly correspond to meaningful shapes or deformations, which is not the case for our construction, and is an assumption we want to avoid. Thus, we propose a construction that is similar to that of the positions subspace described in Section 4.

Again, we choose k' approximately equidistant sample vertices on the mesh and construct weight functions $\mathbf{w}_j$ with a limited support around each of the samples. For efficiency reasons, we use the first k' vertices of the k vertex samples from the construction of $\mathbf{U}$ (in all our examples we have $k' < k$). Thus, the corresponding weight functions can be also be reused, given that the radius was chosen large enough such that this subset of weight functions still covers all vertices with sufficiently many non-zero weights. The weight functions are then interpolated at the relevant elements for the current constraint type (*i.e.* at the triangles for surface strain constraints, tetrahedrons for volume preservation constraints, etc.) and then normalized to sum to one at each element.

We then evaluate all constraint projections $\mathbf{p}_i = \mathbf{p}_i(\mathbf{q}_0)$ of the rest shape $\mathbf{q}_0$ of the mesh. Note that usually we have $\mathbf{p}_i(\mathbf{q}_0) = \mathbf{S}_i \mathbf{q}_0$ for the rest-shape (such that no inner forces act on it). Note that for all types of constraints that have been introduced so far, these projections are geometric quantities such as mean curvature vectors (for bending energies), transformation matrices (for strain or volume preservation energies) or simply edges of the mesh (for spring energies). We can now think of these quantities as being attached to the corresponding elements of the mesh. By replacing the role of the vertex positions $\mathbf{q}_i$ with the constraint projections $\mathbf{p}_i$, we can construct a subspace $\mathbf{V}$ from the degrees of freedom offered by the weighted affine transformations of the quantities $\mathbf{p}_i$ according to transformations chosen at each of the k' handles and weights defined at the elements. The implicit assumption in this construction is that the *transformations of* the rest-shape's constraint projections vary smoothly across the mesh when considering deformations of the shape (note that we do *not* assume that the constraint projections themselves vary smoothly, which is not the case). The explicit construction of this space is explained in Appendix C. As a linear subspace, we are unable to guarantee that all constraint projections of the approximated vectors will be on the constraint manifold, but

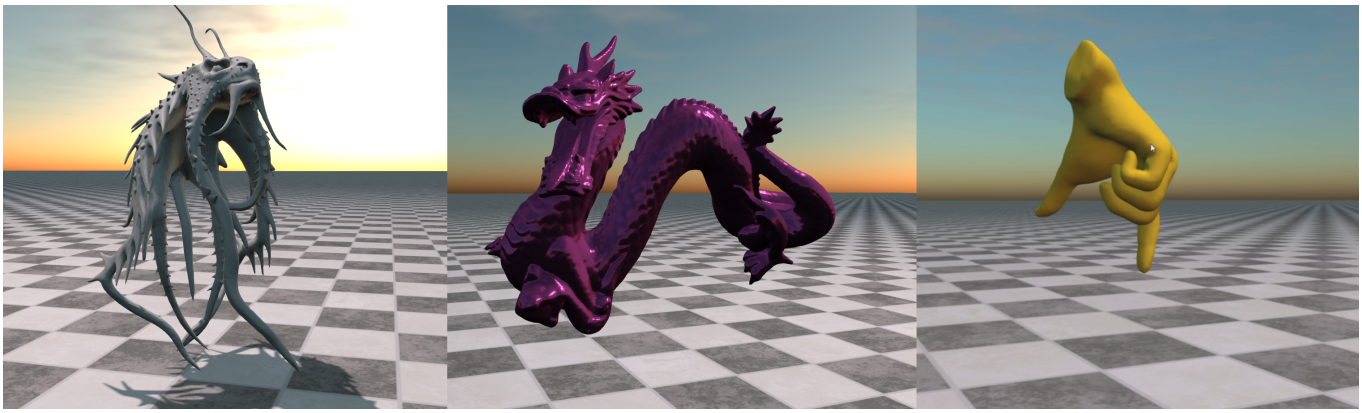

Fig. 2. Compilation of frames from three of the simulations shown in the **supplementary video**. The squid simulation (440k vertices, 1.6 million tetrahedrons) runs at 37 fps (including rendering and ground collision handling), the rest of the examples run at least at 60 fps.

by solving a fitting problem into this space (see next paragraph) we invoke this property in a least squares sense.

To avoid redundancy in the fitting problem and take into account the geometry of the mesh, we perform a weighted PCA on the space constructed above and only use the most significant half of the principal component vectors as the final basis $\mathbf{V}$. As weights we use the length, area or volume associated to the elements of this constraint type.

Constraint projections fitting. Since $\mathbf{V}$ is not built from snapshots of constraint projections vectors, but as a general purpose space, it is unsuited to compute a direct interpolation from a very small number of carefully selected elements as suggested by the DEIM method [Chaturantabut and Sorensen 2010]. Instead, we solve a least-squares fitting problem from considerably more entries in the vector of constraint projections than there are basis vectors in $\mathbf{V}$. Therefore we choose $s \gg k'$ approximately equidistant constraints via the furthest point sampling method, starting with k' constraints next to the k' vertex samples used in the construction of $\mathbf{V}$. In our examples we choose between 800 to 1000 constraint samples, contributing 9 entries in the constraint projections vector each, whereas the dimension of the constructed constraint projections subspace is between 300 and 420, see Table 1.

Let $\mathbf{J} \in \mathbb{R}^{p \cdot s \times p \cdot e}$ be the matrix that maps a vector containing all constraint projections to a vector containing only the s sampled constraints' projections (e is the total number of constraints, and p denotes the number of rows of each constraint projection $\mathbf{p}_i$), and let

$$\mathbf{S}^T = \begin{pmatrix} \lambda_i \mathbf{S}_1^T & \lambda_i \mathbf{S}_2^T & \cdots \end{pmatrix} \tag{7}$$

be the summation matrix, which maps vectors $\mathbf{p} \in \mathbb{R}^{pe \times 3}$ of stacked constraint projections $\mathbf{p}_i$ to the term $\sum_i \lambda_i \mathbf{S}_i^T \mathbf{p}_i$.

During precomputation a list of all vertices that appear in the evaluation of the constraint projections of the s sampled constraints is assembled. In the local phase, the first step is to evaluate these vertices via $\mathbf{q}_i = \mathbf{U}_i \tilde{\mathbf{q}}$ for the current subspace coordinates $\tilde{\mathbf{q}}$ of the system. This can be done in parallel and the vector-vector product above is sparse (due to our specific subspace construction). Then, for each sampled constraint i, we evaluate the corresponding constraint projection $\mathbf{p}_i(\mathbf{q})$ for the current deformation $\mathbf{q}$. We stack these in vectors $\mathbf{p}_{\text{partial}} \in \mathbb{R}^{ps \times 3}$. Thereafter, the least-squares fitting problem is solved by minimizing the residual

$$\left\| \mathbf{J}\mathbf{V}\tilde{\mathbf{p}} - \mathbf{p}_{\text{partial}} \right\|_F^2 \tag{8}$$

which amounts to solving the three uncoupled linear systems

$$\mathbf{V}^T\mathbf{J}^T\mathbf{J}\mathbf{V}\tilde{\mathbf{p}} = \mathbf{V}^T\mathbf{J}^T\mathbf{p}_{\text{partial}} \tag{9}$$

for each coordinate in parallel. Finally, $\tilde{\mathbf{p}}$ is directly mapped to the approximation of the term $\mathbf{U}^T \sum_i \lambda_i \mathbf{S}_i^T \mathbf{p}_i(\mathbf{q})$ via $\mathbf{r} = \mathbf{U}^T\mathbf{S}^T\mathbf{V}\tilde{\mathbf{p}}$, The matrix $\mathbf{U}^T\mathbf{S}^T\mathbf{V}$ can be precomputed and is small (compared to the original dimension of the system).

6 HYPER-REDUCED PROJECTIVE DYNAMICS

After the subspaces $\mathbf{U}$ and $\mathbf{V}$ for vertex positions and constraint projections have been constructed, the sampled constraints have been selected and the fitting problem has been set up, we are able to use the hyper-reduced variant of Projective Dynamics listed in Algorithm 2. There we use the notation $\tilde{\cdot}$ for quantities projected into the subspace, such as $\tilde{f}_{\text{ext}}$ being the external forces projected into the space spanned by $\mathbf{U}$.

The important differences to the original Projective Dynamics method are the modifications to the local and global steps: instead of evaluating all constraint projections, we evaluate a small subset and solve a least-squares fitting problem. Also, instead of solving a global system in the size of the number of vertices, we solve the reduced system to obtain subspace coordinates.

For brevity, Algorihtm 2 does not include the possibility of adding additional, fully evaluated constraint projections to the system. This is entirely possible and was used in our experiments, for example when 'hanging' objects using a few position constraints. These are all evaluated in the local step and added to the right hand side of the global step individually, as in the unreduced method.

If the mesh needs to be displayed at the end of the time step, the updated subspace coordinates need to be mapped to full coordinates. For complex geometries (>15k vertices), we perform this step on the GPU. Additionally, partial position updates are performed using the

Table 1. Data for the experiments shown in the figures and the supplementary video. See Section 7 for further details.

Name	Glove (Fig. 2)	Armadillo (Fig. 5 & 6)	Armadillo (large subspace)	Elephant (Fig. 1)	Dragon (Fig. 2)	Squid (Fig. 2)
Constraints type	Bending & Tri. strain	Tet. strain	Tet. strain	Tri. & Tet. strain	Tet. strain	Tet. strain
# vertices n	24370	19064	19064	52812	196577	439061
# iterations per frame	10	10	10	10	10	10
Position Subspace Dimension	540	1440	3960	1200	1200	2160
Constraint Projections Subspace Dimension	300	360	360	360	420	360
# constraints	73106	71289	71289	277363	733649	1664908
s = # evaluated constraint projections	$2 \cdot 1000$	970	970	$2 \cdot 800$	1000	1000
(# of evaluated vertices)	5771	3688	3688	3789	2741	3931
Precomputation (in seconds)	**8.23**	**6.20**	**16.23**	**21.19**	**63.10**	**189.54**
Local step (in microseconds)	546	337	660	612	643	483
Global step (in microseconds)	48	40	256	148	35	74
Partial update v. pos. (in microseconds)	435	412	678	468	394	524
fps (fps incl. display)	**94 (89)**	**120 (112)**	**63 (60)**	**81 (62)**	**89 (60)**	**116 (37)**
Precomputation unreduced (in seconds)	0.56	0.29	0.29	1.368	11.83	12.57
fps unreduced	2.39	3.7	3.7	0.68	0.20	0.08

GPU, i.e. mapping the subspace coordinates to the positions of only those vertices that appear in sampled constraint projections. Other than that we implemented this method using only CPU operations. More details on our implementation can be found in Appendix A. There, we also specify how we perform the collision handling and user interaction mentioned in Algorithm 2.

Relation to reduced finite element methods. For the presented hyper-reduction of Projective Dynamics, we focused on a specific class of potentials, for which the method is particularly efficient. This limits the type of material response our method can achieve. For example, we can simulate elastic solids with material parameters controlled by Lamé parameters, such as the materials presented in [Chao et al. 2010], but we cannot handle general nonlinear materials. Reduced finite element schemes like [An et al. 2008; von Tycowicz et al. 2013; Wu et al. 2015] can deal with various types of materials. Nevertheless, the benefits that we gain from our hyper-reduction scheme for Projective Dynamics are:

- The matrix for the global step is sparse and constant, hence a sparse factorization can be computed once and re-used throughout the whole simulation.
- The iterations for time-stepping do not involve a line search.
- For both linear solves appearing in our algorithm (constraint projections fitting and global step), the x, y, and z-coordinates are decoupled; hence systems of smaller size are solved in parallel.

As a result, we observe that our approach can achieve real-time rates of 60 fps in subspaces that are roughly one order of magnitude larger than what is reported for recent reduced finite element schemes [von Tycowicz et al. 2013; Wu et al. 2015]. The larger subspaces allow us to gracefully handle features like user interaction, ground collisions, and drastic changes of material stiffness and volume to the real-time simulation. Another feature of the proposed scheme is the fast precomputation stage, see Table 1. In addition, the method profits from the robustness and generality of Projective Dynamics.

Algorithm 2 Hyper-Reduced Projective Dynamics

Input: Current subspace coordinates $\tilde{\mathbf{q}}(t)$ and subspace velocities $\tilde{\mathbf{v}}(t)$
Let $\tilde{\mathbf{s}} = \tilde{\mathbf{q}}(t) + h\tilde{\mathbf{v}}(t) + h^2\mathbf{U}^T\mathbf{M}^{-1}\mathbf{U}\tilde{\mathbf{f}_{\text{ext}}}$
Modify $\tilde{\mathbf{s}}$ according to user interaction and collision constraints.
Let $\hat{\tilde{\mathbf{q}}} = \tilde{\mathbf{s}}$
for i = 1 **to** *numIterations* **do**
  Local step:
    Evaluate required vertex positions $\hat{\mathbf{q}}_{\text{partial}} = \mathbf{U}_{\text{partial}}\hat{\tilde{\mathbf{q}}}$
    Evaluate sampled constraint projections $\mathbf{p}_{\text{partial}}$
    Solve the fitting problem
      $\mathbf{V}^T\mathbf{J}^T\mathbf{J}\mathbf{V}\tilde{\mathbf{p}} = \mathbf{V}^T\mathbf{J}^T\mathbf{p}_{\text{partial}}$
    Build the right hand side term $\mathbf{r} := \mathbf{U}^T\mathbf{S}^T\mathbf{V}\tilde{\mathbf{p}}$
  Global step: Solve
    $\mathbf{U}^T\left(\frac{\mathbf{M}}{h^2} + \sum_i \lambda_i \mathbf{S}_i^T\mathbf{S}_i\right)\mathbf{U}\tilde{\mathbf{q}} = \mathbf{U}^T\frac{\mathbf{M}}{h^2}\mathbf{U}\tilde{\mathbf{s}} + \mathbf{r}$
  Let $\hat{\tilde{\mathbf{q}}} = \tilde{\mathbf{q}}$
end for
Output: Updated subspace coordinates $\tilde{\mathbf{q}}(t+h) = \hat{\tilde{\mathbf{q}}}$ and velocities $\tilde{\mathbf{v}}(t+h) = (\tilde{\mathbf{q}}(t+h) - \tilde{\mathbf{q}}(t))/h$

7 RESULTS

We implemented our Hyper-Reduced Projective Dynamics method for the simulation of triangle and tetrahedral meshes, supporting spring, strain (triangle and tetrahedron), bending, and position constraints, as well as collision handling and user interaction. Details on the implementation can be found in Appendix A.

Many examples, comparisons and evaluations of real time simulations using our hyper-reduced Projective Dynamics method can be found in the supplementary video. There we demonstrate that our method is capable of simulating highly detailed meshes with more than 400k vertices in real-time, while handling collisions and user interactions.

For a general impression of the quality of the hyper-reduced simulations, we show a comparison of simulating the armadillo dropping on the floor using the full method and our hyper-reduced

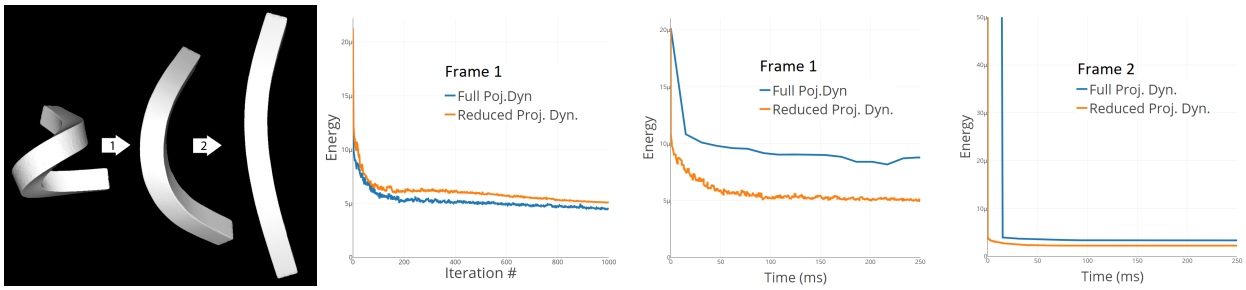

Fig. 3. Progress plots displaying the evolution of the objective (2) during minimization using the local/global solver for the full and the reduced problem. Large time steps are taken and the resulting frames are shown on the left. For the first time step, the objective value versus iterations *and* time are shown. For the second timestep, objective value versus time are shown.

method (the reduction parameters are listed in Table 1). Three frames of each simulation are shown in Figure 5, the full sequences are shown in the supplementary video. We choose a very small time step to highlight differences in both simulations: in the full simulation (left), individual fingers and ears show richer dynamics, but we found that these differences are hard to spot once we choose a more realistic time step (shown in the video). While the full method offers higher detail in the finer dynamics, it results in 3.7 fps and is unsuited for real-time applications.

In the supplementary video we additionally show the dropping armadillo sequence when we restrict the simulation to the subspace, but do not approximate the forces. The resulting dynamics are coarser than in the full simulation, but finer than in the hyper-reduced simulation. This shows that our subspace offers enough degrees of freedom to enable rich dynamics. However, not employing a force approximation scheme results in the same FPS as the full simulation: while the computation time of the global step is significantly reduced (in this example from 3100 microseconds to 70 microseconds), the cost of the evaluation of the right hand side term dominates the iteration times for both approaches. Moreover, evaluating the term in the subspace-only-reduction requires evaluating the current positions $\mathbf{q} = \mathbf{U}\tilde{\mathbf{q}}$ in *every iteration* as well as multiplying the vector $\sum_i \lambda_i \mathbf{S}_i^T \mathbf{p_i}$ by $\mathbf{U}^T$. This means that the cost for the local step is larger in the subspace-only-reduction, than in the full simulation. This shows the importance of employing a force approximation along with a dimension reduction.

Table 1 contains data for the experiments shown in Figures 1, 2 and 6 (these experiments are all shown as sequences in the supplementary video as well). A full explanation for each line of the table is given in the next paragraph. Thereafter, we discuss precomputation and online timings and the tradeoff between computation timings and approximation errors when comparing simulations using our reduced method to full Projective Dynamics simulations.

Table details. In Table 1 we show, for the experiments illustrated in the figures and the supplementary video, the type(s) of constraints used, the number of vertices of the mesh, the position subspace dimension (*i.e.* the degrees of freedom for the vertex positions), the original number of constraints, the number s of constraints that we evaluate for the constraint projections fitting method and the number of vertices whose position needs to be available when computing these constraint projections. We then list the resulting precomputation times in seconds, both for our method as well as for the unreduced method (factorization of the l.h.s. matrix), followed by the computation times (in *micro*seconds) of the local and global step of one iteration of the hyper-reduced method, as well as the time it takes to evaluate the vertex positions of the vertices required to evaluate the sampled constraints. The resulting fps for our hyper-reduced method (using ten iterations per frame) are shown with and without taking into account that all vertex positions have to be evaluated once per frame if the mesh is rendered each frame. Finally, we show the fps of the unreduced method. In the following two paragraphs, we discuss the precomputation and online timings in more detail.

Precomputation times. The setup of the Hyper-Reduced Projective Dynamics method encompasses

- the construction of the positions subspace, which includes
  - sampling k approximately equidistant vertex samples
  - constructing the k weight functions centered at those vertices
- the preparation of the constraint projections fitting method, which includes
  - constructing the subspace of unassembled constraint projections from k' approximately equidistant vertex samples
  - a PCA of that space
  - choosing s sampled constraints
  - the evaluation of all constraint projections of the rest shape
  - the construction and factorization of the left hand side matrix of the constraint projections fitting problem
- the construction and factorization of the left hand side matrix for the global step.

Table 2. Normalized L_2 errors when projecting the frames of the full simulation shown in Figure 5, left, to subspaces of three different sizes constructed with our method.

Subspace dimension	Mean Error	Max. Error
360	0.0269	0.0513
1440	0.0106	0.0228
3960	0.0057	0.0112

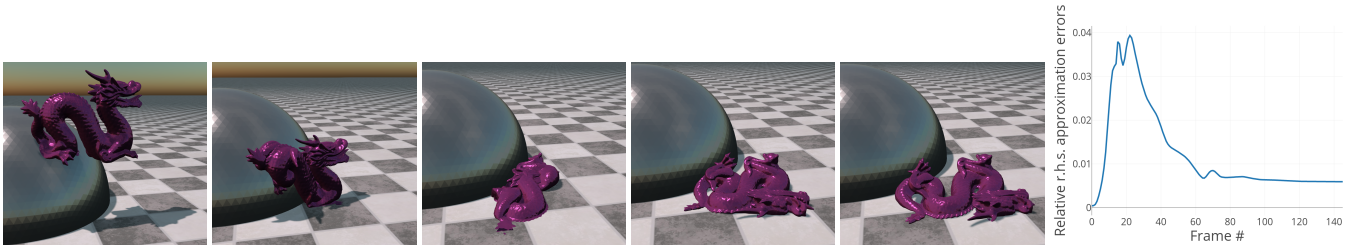

Fig. 4. The dragon mesh is dropped, slides along a sphere and lands on the floor in a heavily deformed state from which it slowly recovers. For each frame of the simulation we plot the relative error of our approximation of the right hand side term from 1000 sampled constraint projections to the fully evaluated term from all 766k constraint projections.

The precomputation timings listed in Table 1 include all of the above steps. The timing depends on the number of vertices n of the original mesh, and the reduction parameters k, k' and s. As the vertex count increases, performing the weighted PCA in the construction of the constraint projections subspace becomes the most time consuming task. The precomputation times measured for the meshes in our experiments (ranging from 19k to 440k vertices) admit most types of applications and would rarely pose severe limitations. Note that all precomputed quantities are suited for arbitrary user interaction, collision constraints and external forces and the precomputation (for a specific mesh) only has to be performed again, when the material type changes, or stiffness weights change in a non-uniform way (*i.e.* by more than a constant factor).

Simulation times. The online phase of our Hyper-Reduced Projective Dynamics method is made up of three steps:

- The first step (not present in the unreduced method), in which we evaluate the positions of the vertices that are required to evaluate the sampled constraint projections in the next iteration, since only subspace coordinates are available at that point.
- The local step, which encompasses evaluating the sampled constraint projections, solving a fitting problem and mapping it to the approximation of the right hand side term required for the global step.
- The global step, in which we solve the reduced linear system.

When the mesh is being displayed after each time integration step, the full vertex positions need to be evaluated once per frame via $\mathbf{q} = \mathbf{U}\tilde{\mathbf{q}}$, which we perform on the GPU for meshes with more than 15k vertices, and on the CPU, in parallel, otherwise. This step becomes the most time-consuming part of the simulation when the mesh resolution is high: for the Squid mesh (Figure 2, left) with 440k vertices, we can compute 116 time steps of the simulation per second (when using ten iterations per time step), but additionally evaluating the full positions $\mathbf{q}$ to render the current state results in 37 fps. For all other examples, including the Dragon mesh with almost 200k vertices, we achieve 60 fps or more, including rendering, collision detection and user interaction. Our hyper-reduced method is able to handle large subspaces while still maintaining high frame rates: for the armadillo mesh we can use a subspace that offers 3960 degrees of freedom and still achieve 60 fps including rendering and collision detection, see Table 1.

Convergence. In order to compare the convergence of both the full and our Hyper-Reduced Projective Dynamics methods, we simulate a volumetric elastic bar whose rest shape is an undeformed block, but is initially set to a heavily bended position, as seen in Figure 3, left. The internal forces lead to an unfolding of the bar and we choose a large time step such that the optimization problem (2) is sufficiently hard. Throughout the minimization we record the energy levels over time of both methods, see Figure 3, right. Both methods show similar behavior, in that they heavily reduce the energy in the first few iterations and then slowly converge to a lower optimum (for the second frame, the energy before the first iteration is out of range of the plot). While our method usually requires more iterations to converge to a lower energy level, it takes less time to do so since the iterations are about 30 times faster. With higher resolutions of the mesh, this speed up factor becomes larger, *e.g.* for the Squid mesh we get a factor of 1440, using the reduction parameters listed in Table 1.

In all examples, we use 10 local/global iterations per time integration step. We find that for our hyper-reduced method the differences between a simulation running with 10 iterations and a fully converged solution are very small since the coupling of material stiffness and iteration count can only be observed at very low iteration counts. In the supplementary video, we provide a visual comparison between a simulation running with 10 iterations and the same simulation running with 1000 iterations.

Approximation errors. When comparing the results of our hyper-reduced method to those of the full Projective Dynamics method, there are two main sources of approximation errors:

(1) Errors due to the reduced degrees of freedom offered by our subspace for vertex positions.
(2) Errors when approximating the constraint projections via our fitting method.

To measure the first type of approximation errors mentioned above, that is, to evaluate the degrees of freedom offered by our subspace, we want to quantify how well deformations of a mesh that is simulated in full detail can be approximated by our vertex position subspace. Therefore, we project each frame of the *full* simulation of a dropping armadillo (Figure 5, left) into subspaces constructed via the method described in Section 4. In Table 2 we list the mean and maximal L_2 differences between the full and the projected shapes of all frames of the simulation for three different subspace sizes (the

original shape was normalized to have L_2 norm 1). The mean error does not include the first few frames of the simulation where the armadillo is still in its rest shape, which results in an approximation error of zero. Note, that even when running the simulation using our hyper-reduced method with the largest of the three subspaces, we still achieve 60 fps including rendering and collision detection (see Table 1).

The second type of approximation errors (approximated constraint projections) is measured as the relative errors between the right hand side term $\sum_i \lambda_i \mathbf{S}_i \mathbf{p}_i(\mathbf{q})$ to its approximation $\mathbf{SV}\tilde{\mathbf{p}}$, acquired from our constraint projections fitting method (see Section 5). We plot these errors for the dropping dragon simulation in Figure 4, where we evaluate only 1000 of the 733649 constraints to approximate the right hand side term.

Generality. One of the reasons that we chose to reduce the Projective Dynamics method is its generality. It is able to handle thin shells, volumetric deformables, spring-based materials, example-based materials, and more. In the supplementary video we show examples for various volumetric deformables (the armadillo, the dragon and the squid), a thin shell mesh (the glove) and a combination of surface strain and volumetric strain (the elephant). Frames of these simulations are shown in Figures 1 and 2. We show that we can handle different types of constraints simultaneously and change the target volume of the tetrahedrons, while limiting the strain on the outer triangles of the mesh during simulation, as seen in Figure 1. This demonstrates the range of flexibility that our hyper-reduced method is able to carry over from the Projective Dynamics method.

Moreover, we obtain additional flexibility via our subspace reduction: the system matrix for the global step is small (in comparison to the size of the full system) but still sparse (because of our specific subspace construction). This enables us to change constraint weights or add and remove constraints in the online phase of the simulation without dropping below 60 frames per second. This is shown in various examples in the supplementary video and in Figure 6, where we change the stiffness parameter in running simulations. For the unreduced method on the armadillo mesh, refactorizing the system matrix takes 108 milliseconds, which prohibits changes to the constraints amid simulation.

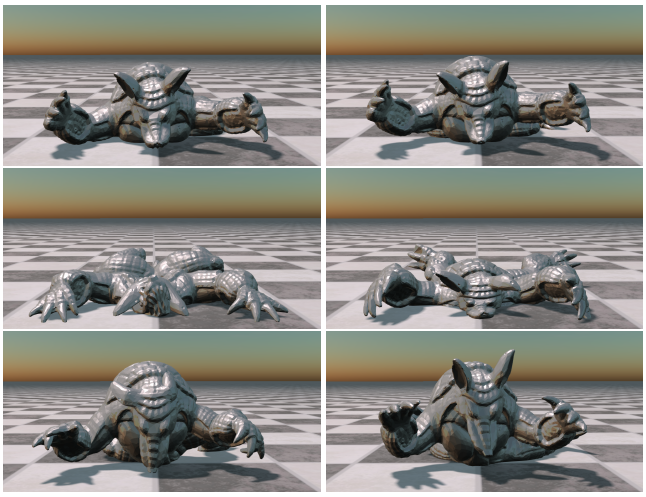

Fig. 5. Visual comparison of simulation results from full Projective Dynamics (left column) and our method using 1440 degrees of freedom (right column) both starting with the same initialization.

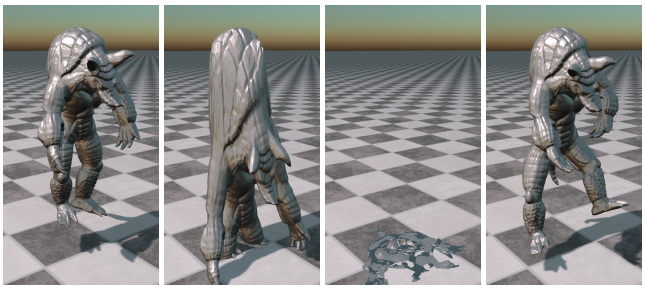

Fig. 6. We are able to interactively change stiffness parameters in a running simulation. From left to right we scale the predefined weights by 1, 0.1, 0 and again 1. This also demonstrates that our method is able to recover from a fully flattened state.

Robustness. Our hyper-reduced method inherits the robustness properties of Projective Dynamics, which is, as an implicit variational integration scheme, unconditionally stable, and has been demonstrated to gracefully handle highly deformed states. In Figure 6 we show that our hyper-reduced method is able to restore a tetrahedral mesh from a completely flattened state. Throughout the experiments shown in the supplementary video we subject the meshes to large external forces, rapid user interaction and collision constraints, and the simulations show stable behavior throughout.

8 CONCLUSION

We presented a method for real-time simulation of deformables that combines the benefits of Projective Dynamics and hyper-reduction techniques in one framework. The resulting scheme is robust, general and efficient. It enables real-time simulation of high-resolution meshes in specifically designed subspaces with up to 4k degrees of freedom while keeping precomputation times low. We provide examples that include deformable solids and shells, user interaction, collision constraints, external forces, varying material stiffness, and recovery from degenerate configurations.

Limitations and challenges. While we handle collisions with rigid objects, one limitation of our current implementation is that self-collisions and collisions between multiple deforming meshes of the deformable object are not handled. It would be interesting to integrate techniques like *collision certificates* [Barbič and James 2010], *bounded-normal trees* [Schvartzman et al. 2009] and *pose-space cubature* [Teng et al. 2014] to the proposed approach.

The subspaces used for positions and constraint projections are not directly suited for simulating cloth, since the assumption of deformations that vary continuously across the mesh is no longer valid: When bending resistance is very low or not simulated at all, sharp creases and noisy features start appearing, which can not be faithfully reconstructed using the presented subspaces. Here, performing full simulations and creating subspaces for positions

and constraint projections from snapshots might prove beneficial. Initial tests showed that such subspaces are very specific to the task that was performed in the full simulation (*i.e.*, how the cloth was grabbed and deformed and what collision constraints and external forces were present) and do not enjoy the generality of our method.

The Projective Dynamics method has been generalized to handle fluids by Weiler et al. [2016], but the loss of any connectivity renders our reduction method unable to handle particles. It is an interesting question whether the reduction can be adjusted such that it can be applied to Projective Fluids.

Lastly, the recent generalizations of Projective Dynamics to nonlinear constitutive materials in [Liu et al. 2017] and [Narain et al. 2016; Overby et al. 2017] offer a direction to extend our hyper-reduced method in a similar fashion.

ACKNOWLEDGMENTS

We would like to thank Christoph von Tycowicz for constructive discussions and his feedback, Leonardo Scandolo for helping with the real-time rendering, and the anonymous reviewers for helpful comments and suggestions.

REFERENCES

Steven S. An, Theodore Kim, and Doug L. James. 2008. Optimizing cubature for efficient integration of subspace deformations. *ACM Trans. Graph.* 27, 5 (2008), 1–10.

Jernej Barbič and Doug L. James. 2005. Real-Time Subspace Integration for St. Venant-Kirchhoff Deformable Models. *ACM Trans. Graph.* 24, 3 (2005), 982–990.

Jernej Barbič and Doug L. James. 2010. Subspace Self-collision Culling. *ACM Trans. Graph.* 29, 4 (2010), 81:1–81:9.

Jernej Barbič, Funshing Sin, and Eitan Grinspun. 2012. Interactive Editing of Deformable Simulations. *ACM Trans. Graph.* 31, 4 (2012), 70:1–70:8.

Maxime Barrault, Yvon Maday, Ngoc Cuong Nguyen, and Anthony T. Patera. 2004. An 'empirical interpolation' method: application to efficient reduced-basis discretization of partial differential equations. *Comptes Rendus Mathematique* 339, 9 (2004), 667–672.

Jan Bender, Dan Koschier, Patrick Charrier, and Daniel Weber. 2014. Position-based simulation of continuous materials. *Computers & Graphics* 44 (2014), 1 – 10.

Jan Bender, Matthias Müller, and Miles Macklin. 2017. Position-Based Simulation Methods in Computer Graphics. In *EUROGRAPHICS 2017 Tutorials*.

Sofien Bouaziz, Sebastian Martin, Tiantian Liu, Ladislav Kavan, and Mark Pauly. 2014. Projective Dynamics: Fusing Constraint Projections for Fast Simulation. *ACM Trans. Graph.* 33, 4 (2014), 154:1–154:11.

Christopher Brandt and Klaus Hildebrandt. 2017. Compressed Vibration Modes of Deformable Bodies. *Computer Aided Geometric Design* 52–53 (2017), 297–312.

Christopher Brandt, Christoph von Tycowicz, and Klaus Hildebrandt. 2016. Geometric Flows of Curves in Shape Space for Processing Motion of Deformable Objects. *Computer Graphics Forum* 35, 2 (2016).

Marcel Campen, Martin Heistermann, and Leif Kobbelt. 2013. Practical Anisotropic Geodesy. *Computer Graphics Forum* 32, 5 (2013), 63–71.

Steve Capell, Seth Green, Brian Curless, Tom Duchamp, and Zoran Popović. 2002. A Multiresolution Framework for Dynamic Deformations. In *Proc. ACM SIGGRAPH/Eurographics Symposium on Computer Animation*. 41–47.

Jeffrey N. Chadwick, Steven S. An, and Doug L. James. 2009. Harmonic shells: a practical nonlinear sound model for near-rigid thin shells. *ACM Trans. Graph.* 28, 5 (2009), 119:1–119:10.

Isaac Chao, Ulrich Pinkall, Patrick Sanan, and Peter Schröder. 2010. A simple geometric model for elastic deformations. *ACM Trans. Graph.* 29 (2010), 38:1–38:6.

Saifon Chaturantabut and Danny C Sorensen. 2010. Nonlinear model reduction via discrete empirical interpolation. *SIAM Journal on Scientific Computing* 32, 5 (2010), 2737–2764.

Xiang Chen, Changxi Zheng, and Kun Zhou. 2017. Example-Based Subspace Stress Analysis for Interactive Shape Design. *IEEE Trans. Vis. and Comp. Graph.* 23, 10 (2017), 2314–2327.

Min Gyu Choi and Hyeong-Seok Ko. 2005. Modal Warping: Real-Time Simulation of Large Rotational Deformation and Manipulation. *IEEE Trans. Vis. Comput. Graphics* 11, 1 (2005), 91–101.

Leonardo Dagum and Ramesh Menon. 1998. OpenMP: An Industry-Standard API for Shared-Memory Programming. *IEEE Comput. Sci. Eng.* 5, 1 (1998), 46–55.

Gilles Debunne, Mathieu Desbrun, Marie-Paule Cani, and Alan H. Barr. 2001. Dynamic Real-time Deformations Using Space &Amp; Time Adaptive Sampling. In *Proc. ACM SIGGRAPH*. 31–36.

Benjamin Gilles, Guillaume Bousquet, Francois Faure, and Dinesh K. Pai. 2011. Frame-based Elastic Models. *ACM Trans. Graph.* 30, 2 (2011), 15:1–15:12.

Gaël Guennebaud, Benoît Jacob, et al. 2010. Eigen v3. http://eigen.tuxfamily.org. (2010).

Fabian Hahn, Bernhard Thomaszewski, Stelian Coros, Robert W. Sumner, Forrester Cole, Mark Meyer, Tony DeRose, and Markus H. Gross. 2014. Subspace clothing simulation using adaptive bases. *ACM Trans. Graph.* 33, 4 (2014), 105:1–105:9.

David Harmon and Denis Zorin. 2013. Subspace integration with local deformations. *ACM Trans. Graph.* 32, 4 (2013), 107:1–107:10.

Klaus Hildebrandt, Christian Schulz, Christoph von Tycowicz, and Konrad Polthier. 2011. Interactive surface modeling using modal analysis. *ACM Trans. Graph.* 30, 5 (2011), 119:1–119:11.

Klaus Hildebrandt, Christian Schulz, Christoph von Tycowicz, and Konrad Polthier. 2012. Interactive spacetime control of deformable objects. *ACM Trans. Graph.* 31, 4 (2012), 71:1–71:8.

Jin Huang, Xiaohan Shi, Xinguo Liu, Kun Zhou, Li-Yi Wei, Shang-Hua Teng, Hujun Bao, Baining Guo, and Heung-Yeung Shum. 2006. Subspace gradient domain mesh deformation. *ACM Trans. Graph.* 25, 3 (2006).

Jin Huang, Yiying Tong, Kun Zhou, Hujun Bao, and Mathieu Desbrun. 2011. Interactive Shape Interpolation through Controllable Dynamic Deformation. *IEEE Trans. Vis. Comput. Graph.* 17, 7 (2011), 983–992.

Alec Jacobson, Ilya Baran, Ladislav Kavan, Jovan Popović, and Olga Sorkine. 2012. Fast Automatic Skinning Transformations. *ACM Trans. Graph.* 31, 4 (2012), 77:1–77:10.

Alec Jacobson, Ilya Baran, Jovan Popovic, and Olga Sorkine. 2011. Bounded biharmonic weights for real-time deformation. *ACM Trans. Graph.* 30, 4 (2011), 78:1–78:8.

Lily Kharevych, Patrick Mullen, Houman Owhadi, and Mathieu Desbrun. 2009. Numerical Coarsening of Inhomogeneous Elastic Materials. *ACM Trans. Graph.* 28, 3 (2009), 51:1–51:8.

Theodore Kim and John Delaney. 2013. Subspace Fluid Re-simulation. *ACM Trans. Graph.* 32, 4 (2013), 62:1–62:9.

Tae-Yong Kim, Nuttapong Chentanez, and Matthias Müller-Fischer. 2012. Long Range Attachments - a Method to Simulate Inextensible Clothing in Computer Games. In *Proc. ACM SIGGRAPH/Eurographics Symposium on Computer Animation*. 305–310.

Petr Krysl, Sanjay Lall, and Jerrold E. Marsden. 2001. Dimensional Model Reduction in Non-linear Finite Element Dynamics of Solids and Structures. *Int. J. Numer. Meth. Eng.* 51 (2001), 479–504.

Siwang Li, Jin Huang, Fernando de Goes, Xiaogang Jin, Hujun Bao, and Mathieu Desbrun. 2014. Space-time Editing of Elastic Motion Through Material Optimization and Reduction. *ACM Trans. Graph.* 33, 4 (2014), 108:1–108:10.

Tiantian Liu, Adam W. Bargteil, James F. O'Brien, and Ladislav Kavan. 2013. Fast Simulation of Mass-spring Systems. *ACM Trans. Graph.* 32, 6 (2013), 214:1–214:7.

Tiantian Liu, Sofien Bouaziz, and Ladislav Kavan. 2017. Quasi-Newton Methods for Real-Time Simulation of Hyperelastic Materials. *ACM Trans. Graph.* 36, 3 (2017), 23:1–23:16.

Miles Macklin and Matthias Müller. 2013. Position Based Fluids. *ACM Trans. Graph.* 32, 4 (2013), 104:1–104:12.

Miles Macklin, Matthias Müller, and Nuttapong Chentanez. 2016. XPBD: Position-based Simulation of Compliant Constrained Dynamics. In *Proc. ACM Motion in Games*. 49–54.

Sebastian Martin, Bernhard Thomaszewski, Eitan Grinspun, and Markus Gross. 2011. Example-based Elastic Materials. *ACM Trans. Graph.* 30, 4 (2011), 72:1–72:8.

Matthias Müller. 2008. Hierarchical Position Based Dynamics. In *Proc. Workshop on Virtual Reality Interaction and Physical Simulation (VRIPHYS)*.

Matthias Müller and Nuttapong Chentanez. 2011. Solid Simulation with Oriented Particles. *ACM Trans. Graph.* 30, 4 (2011), 92:1–92:10.

Matthias Müller, Nuttapong Chentanez, Tae-Yong Kim, and Miles Macklin. 2014. Strain Based Dynamics. In *Proc. ACM SIGGRAPH/Eurographics Symposium on Computer Animation*. 149–157.

Matthias Müller, Bruno Heidelberger, Marcus Hennix, and John Ratcliff. 2007. Position Based Dynamics. *J. Vis. Comun. Image Represent.* 18, 2 (2007), 109–118.

Matthias Müller, Bruno Heidelberger, Matthias Teschner, and Markus Gross. 2005. Meshless Deformations Based on Shape Matching. In *Proc. ACM SIGGRAPH*. 471–478.

Przemyslaw Musialski, Thomas Auzinger, Michael Birsak, Michael Wimmer, and Leif Kobbelt. 2015. Reduced-order Shape Optimization Using Offset Surfaces. *ACM Trans. Graph.* 34, 4 (2015), 102:1–102:9.

Rahul Narain, Matthew Overby, and George E. Brown. 2016. ADMM ⊇ Projective Dynamics: Fast Simulation of General Constitutive Models. In *Proc. ACM SIGGRAPH/Eurographics Symposium on Computer Animation*. 21–28.

Matthieu Nesme, Paul G. Kry, Lenka Jeřábková, and François Faure. 2009. Preserving Topology and Elasticity for Embedded Deformable Models. *ACM Trans. Graph.* 28, 3 (2009), 52:1–52:9.

Thomas Neumann, Kiran Varanasi, Stephan Wenger, Markus Wacker, Marcus Magnor, and Christian Theobalt. 2013. Sparse Localized Deformation Components. *ACM*

Trans. Graph. 32, 6 (2013), 179:1–179:10.
John Nickolls, Ian Buck, Michael Garland, and Kevin Skadron. 2008. Scalable Parallel Programming with CUDA. *Queue* 6, 2 (2008), 40–53.
Matthew Overby, George E. Brown, Jie Li, and Rahul Narain. 2017. ADMM ⊇ Projective Dynamics: Fast Simulation of Hyperelastic Models with Dynamic Constraints. *IEEE Trans. Vis. and Comp. Graph.* 23, 10 (2017), 2222–2234.
Zherong Pan, Hujun Bao, and Jin Huang. 2015. Subspace Dynamic Simulation Using Rotation-strain Coordinates. *ACM Trans. Graph.* 34, 6 (2015), 242:1–242:12.
Alex Pentland and John Williams. 1989. Good vibrations: modal dynamics for graphics and animation. In *Proc. of ACM SIGGRAPH*. 215–222.
Olivier Rémillard and Paul G. Kry. 2013. Embedded Thin Shells for Wrinkle Simulation. *ACM Trans. Graph.* 32, 4 (2013), 50:1–50:8.
Alec R. Rivers and Doug L. James. 2007. FastLSM: Fast Lattice Shape Matching for Robust Real-time Deformation. *ACM Trans. Graph.* 26, 3 (2007).
Christian Schulz, Christoph von Tycowicz, Hans-Peter Seidel, and Klaus Hildebrandt. 2014. Animating Deformable Objects using Sparse Spacetime Constraints. *ACM Trans. Graph.* 33, 4 (2014), 109:1–109:10.
Sara C. Schvartzman, Jorge Gascón, and Miguel A. Otaduy. 2009. Bounded Normal Trees for Reduced Deformations of Triangulated Surfaces. In *Proc. ACM SIGGRAPH/Eurographics Symposium on Computer Animation*. 75–82.
Jos Stam. 2009. Nucleus: Towards a unified dynamics solver for computer graphics. In *Proc. IEEE International Conference on Computer-Aided Design and Computer Graphics*. 1–11.
Yun Teng, Miguel A. Otaduy, and Theodore Kim. 2014. Simulating Articulated Subspace Self-contact. *ACM Trans. Graph.* 33, 4 (2014), 106:1–106:9.
Philipp von Radziewsky, Elmar Eisemann, Hans-Peter Seidel, and Klaus Hildebrandt. 2016. Optimized subspaces for deformation-based modeling and shape interpolation. *Computers & Graphics* 58 (2016), 128 – 138.
Christoph von Tycowicz, Christian Schulz, Hans-Peter Seidel, and Klaus Hildebrandt. 2013. An Efficient Construction of Reduced Deformable Objects. *ACM Trans. Graph.* 32, 6 (2013), 213:1–213:10.
Christoph von Tycowicz, Christian Schulz, Hans-Peter Seidel, and Klaus Hildebrandt. 2015. Real-time Nonlinear Shape Interpolation. *ACM Trans. Graph.* 34, 3 (2015), 34:1–34:10.
Huamin Wang. 2015. A Chebyshev Semi-iterative Approach for Accelerating Projective and Position-based Dynamics. *ACM Trans. Graph.* 34, 6 (2015), 246:1–246:9.
Yu Wang, Alec Jacobson, Jernej Barbič, and Ladislav Kavan. 2015. Linear Subspace Design for Real-time Shape Deformation. *ACM Trans. Graph.* 34, 4 (2015), 57:1–57:11.
Marcel Weiler, Dan Koschier, and Jan Bender. 2016. Projective Fluids. In *Proc. ACM Motion in Games*. 79–84.
Chris Wojtan and Greg Turk. 2008. Fast Viscoelastic Behavior with Thin Features. *ACM Trans. Graph.* 27, 3 (2008), 47:1–47:8.
Xiaofeng Wu, Rajaditya Mukherjee, and Huamin Wang. 2015. A Unified Approach for Subspace Simulation of Deformable Bodies in Multiple Domains. *ACM Trans. Graph.* 34, 6 (2015), 241:1–241:9.
Hongyi Xu, Yijing Li, Yong Chen, and Jernej Barbic. 2015. Interactive Material Design Using Model Reduction. *ACM Trans. Graph.* 34, 2 (2015), 18:1–18:14.
Yin Yang, Dingzeyu Li, Weiwei Xu, Yuan Tian, and Changxi Zheng. 2015. Expediting Precomputation for Reduced Deformable Simulation. *ACM Trans. Graph.* 34, 6 (2015), 243:1–243:13.
Y. Yang, W. Xu, X. Guo, K. Zhou, and B. Guo. 2013. Boundary-Aware Multidomain Subspace Deformation. *IEEE Trans. Vis. and Comp. Graph.* 19, 10 (2013), 1633–1645.

A IMPLEMENTATION DETAILS

We implemented our Hyper-Reduced Projective Dynamics method using C++, specifically the *Eigen* library [Guennebaud et al. 2010] to handle all linear algebra operations, OpenMP [Dagum and Menon 1998] to handle the parallel execution of constraint projections and CUDA [Nickolls et al. 2008] when mapping reduced coordinates directly to vertex positions in OpenGL buffers (this is only done for meshes with more than 100k vertices). Constraint projections were implemented as detailed in [Bouaziz et al. 2014], with the exception of the bending projection, where we additionally ensure that the dot product between the outer normal of a fixed adjacent triangle and the mean curvature vector keeps the same sign throughout the simulation (this prevents the mesh from permanently inverting along edges where heavy buckling occurs).

User interaction. Instead of using position constraints to handle user interactions, we simply modify the vector $\mathbf{s}$ (which can be interpreted as the desired positions in the next time step, disregarding inner forces), by moving vertices close to the mouse parallel to the camera's viewing plane when click and drag actions are performed. We ignore vertices whose positions are not being evaluated (*i.e.* are not part of the sampled constraints), such that this operation remains independent of the full resolution as well. We then obtain the subspace vector $\tilde{\mathbf{s}}$ by interpolating the positions of all vertices on sampled constraints into the subspace. This method of handling user interaction also prevents us from having to introduce inactive position constraints which unnaturally change the system's behavior.

Collision handling. In presence of collision constraints, one tries to minimize the energy stated in equation (2) subject to inequality constraints on the vertex positions, which come from anticipated collisions of the mesh with static objects. To circumvent both the detection of collisions for every vertex of the mesh in every local/global iteration, as well as adding temporary inequality constraints to the global step solve, we employ the following approximation of this optimization: In every frame of the simulation, once the reduced coordinates of the desired vertex positions $\tilde{\mathbf{s}}$ have been evaluated, we check, for one vertex at each sampled constraint, if there are violated collisions for this vertex and if so, compute the projection of this vertex to a collision free position. Note that the actual positions of these vertices needed to be evaluated anyway, since they are located at sampled constraints. In our reference implementation we used a simple projection method to obtain collision free positions for the vertices and to introduce repulsion and friction, we scale the tangential velocities and normal velocities to the collision plane of vertices that were not collision free by constant factors. This keeps the complexity all computation steps independent of the mesh resolution and offers an efficient and visually convincing way to let the mesh interact with static objects. Examples for meshes being dropped on a floor or colliding with static spheres can be found in the supplementary video. Collisions with thin features and sharp edges cannot be captured well by our approach, since the subset of vertices for which we check and resolve collisions is too coarse to support them. More involved collision resolving strategies and models for friction and repulsion can of course be applied, and, if desired, they can exploit the benefit of handling these for only a subset of the vertices and interpolating the effect via a projection to the subspace.

B BLEND-SKINNING SUBSPACE CONSTRUCTION

Here we detail the construction of a subspace that mimics the degrees of freedom available in linear blend-skinning techniques. Given k handles with associated weights w_i^j per vertex and handle, a skinning transformation of a rest shape $\mathbf{q}_0 \in \mathbb{R}^{n\times 3}$ is obtained via

$$\begin{pmatrix} \hat{q}_i^x \\ \hat{q}_i^y \\ \hat{q}_i^z \end{pmatrix} = \sum_j w_i^j \mathbf{A}_j \begin{pmatrix} q_i^x \\ q_i^y \\ q_i^z \\ 1 \end{pmatrix}, \tag{10}$$

where the $\mathbf{A}_j$ are the k chosen affine transformations (3×4 matrices) for each handle and (q_i^x, q_i^y, q_i^z) is the i-th row of $\mathbf{q}$. From this, one can deduce the subspace matrix

$$\mathbf{U}_j = \begin{pmatrix} w_0^j & q_0^x \cdot w_0^j & q_0^y \cdot w_0^j & q_0^z \cdot w_0^j \\ w_1^j & q_1^x \cdot w_1^j & q_1^y \cdot w_1^j & q_1^z \cdot w_1^j \\ \cdots & & & \end{pmatrix} \tag{11}$$

$$\mathbf{U} = (\mathbf{U}_1 \mid \cdots \mid \mathbf{U}_k) \tag{12}$$

and interpret the entries of the affine transformations as subspace coordinates

$$\tilde{\mathbf{q}} = \begin{pmatrix} \mathbf{A}_0^T \\ \cdots \\ \mathbf{A}_{k-1}^T \end{pmatrix} \tag{13}$$

Then, the transformations (10) can be concisely written as $\hat{\mathbf{q}} = \mathbf{U}\tilde{\mathbf{q}}$. In effect this means that for each handle, or set of weights, we get 4 subspace vectors and 12 degrees of freedom, as the subspace naturally decouples x, y and z coordinates, which goes along well with the decoupled system of the global step of Projective Dynamics.

C SUBSPACE FOR CONSTRAINT PROJECTIONS

The construction above can be extended to subspaces for vectors of unassembled constraint projections. Here we assume the weights w_i^j are defined at the elements associated to the constraints for which this construction is used (*e.g.* at the tetrahedrons for volume preservation constraints). In addition, we have the evaluated constraint projections $\mathbf{p}_i(\mathbf{q}_0) \in \mathbb{R}^{p\times3}$ of the rest shape at each element, which now replace the role of the vertex positions. That is, the subspace coordinates are given by the entries of k' transformations matrices $\mathbf{A}_j \in \mathbb{R}^{3\times4}$, from which we get new constraint projections via the transformations

$$\hat{\mathbf{p}}_i = \sum_j w_i^j \mathbf{A}_j \begin{pmatrix} \mathbf{p}_i \\ \mathbf{1} \end{pmatrix}, \tag{14}$$

where $\begin{pmatrix} \mathbf{p}_i \\ \mathbf{1} \end{pmatrix}$ is simply the $(p+1)\times3$ matrix formed by attaching a row of ones to $\mathbf{p}_i$. Let

$$\mathbf{p}_i = \begin{pmatrix} p_i^{x,0} & p_i^{y,0} & p_i^{z,0} \\ \cdots & & \\ p_i^{x,p-1} & p_i^{y,p-1} & p_i^{z,p-1} \end{pmatrix} \tag{15}$$

Then the subspace matrix $\mathbf{V}$ is defined as follows:

$$\mathbf{V}_j = \begin{pmatrix} w_0^j & p_0^{x,0} \cdot w_0^j & p_0^{y,0} \cdot w_0^j & p_0^{z,0} \cdot w_0^j \\ \cdots & & & \\ w_0^j & p_0^{x,p-1} \cdot w_0^j & p_0^{y,p-1} \cdot w_0^j & p_0^{z,p-1} \cdot w_0^j \\ w_1^j & p_1^{x,0} \cdot w_1^j & p_1^{y,0} \cdot w_1^j & p_1^{z,0} \cdot w_1^j \\ \cdots & & & \end{pmatrix} \tag{16}$$

$$\mathbf{V} = \left(\mathbf{V}_1 \mid \cdots \mid \mathbf{V}'_k\right) \tag{17}$$

That is, $V \in \mathbb{R}^{ep\times4k'}$, where e is the number of elements associated to the constraint projections, p is the size of the constraints and k' was the chosen number of handles. Then, we can write the transformations (14) as

$$\begin{pmatrix} \hat{\mathbf{p}}_0 \\ \cdots \\ \hat{\mathbf{p}}_e \end{pmatrix} = \mathbf{V} \begin{pmatrix} \mathbf{A}_0^T \\ \cdots \\ \mathbf{A}_{k'-1}^T \end{pmatrix} \tag{18}$$