Efficient Incremental Potential Contact for Actuated Face Simulation

Bo Li*
ETH Zurich
Switzerland
bolibo@ethz.ch

Lingchen Yang*
ETH Zurich
Switzerland
lingchen.yang@inf.ethz.ch

Barbara Solenthaler
ETH Zurich
Switzerland
solenthaler@inf.ethz.ch

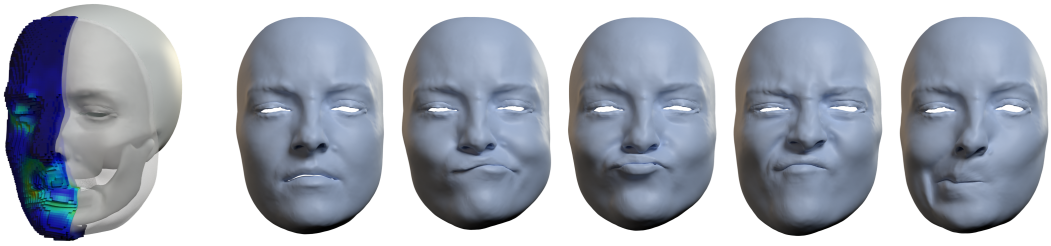


Figure 1: We model the human face as a soft body and simulate its deformation using Finite Element Method driven by the actuation of muscles. With the introduction of collision handling, we are capable of simulating challenging expressions that may result in unrealistic self-intersections if not solved properly.

ABSTRACT

We present a quasi-static finite element simulator for human face animation. We model the face as an actuated soft body, which can be efficiently simulated using Projective Dynamics (PD). We adopt Incremental Potential Contact (IPC) to handle self-intersection. However, directly integrating IPC into the simulation would impede the high efficiency of the PD solver, since the stiffness matrix in the global step is no longer constant and cannot be pre-factorized. We notice that the actual number of vertices affected by the collision is only a small fraction of the whole model, and by utilizing this fact we effectively decrease the scale of the linear system to be solved. With the proposed optimization method for collision, we achieve high visual fidelity at a relatively low performance overhead.

CCS CONCEPTS

• **Computing methodologies** → **Physical simulation**.

KEYWORDS

Physically based simulation, Projective Dynamics, Incremental Potential Contact, Collision Handling

*Contributed equally to this work.

SA Technical Communications '23, December 12–15, 2023, Sydney, NSW, Australia

ACM ISBN 979-8-4007-0314-0/23/12…$15.00
https://doi.org/10.1145/3610543.3626161

ACM Reference Format:
Bo Li, Lingchen Yang, and Barbara Solenthaler. 2023. Efficient Incremental Potential Contact for Actuated Face Simulation. In *SIGGRAPH Asia 2023 Technical Communications (SA Technical Communications '23), December 12–15, 2023, Sydney, NSW, Australia.* ACM, New York, NY, USA, 4 pages. https://doi.org/10.1145/3610543.3626161

1 INTRODUCTION

Various deformable entities, such as human facial structures, can be viewed as soft bodies whose motion is governed by the underlying muscle activation. Yang et al. [Yang et al. 2022] introduced a neural model that implicitly represents muscle actuation as a function of spatial coordinates, which drives the soft body simulation using Projective Dynamics (PD) [Bouaziz et al. 2014]. Yet, this simulation framework does not account for collisions, an aspect which is crucial to realistic animation. For example, contact is ubiquitous around the mouth (see Figure 1, right part), and the inter-penetration between the upper and lower lips would generate implausible artifacts.

Recently, Incremental Potential Contact (IPC) [Li et al. 2020] has gained much popularity for contact modeling. It could achieve large-scale intersection-free deformations. In this work, we integrate IPC into the simulator by Yang et al. Although the integration may seem simple at first glance, there are inherent incompatibilities between IPC and PD. The efficiency of PD is inherited from its reuse of the pre-factorized stiffness matrix for solving a fixed linear system, but a straightforward introduction of the time-varying Hessian matrix of IPC could negate this benefit.

The original collision handling method in the PD framework projects penetrating vertices onto the closest surface and employs spring-like energy terms to redirect them to the collision-free state. However, in addition to slight penetration, it suffers from sticking artifacts since the system only generates repulsion when the penetration really happens. To address this issue, Lan et al. [Lan et al. 2022] proposed a "barrier projection" method that strategically sets the target position for collision vertices. It models the

target position of a vertex after rebounding from the barrier by utilizing its velocity information. However, this approach is unsuitable for quasi-static simulations where the concept of velocity does not exist, leading to a reversion to the standard PD method. On the other spectrum, Wang et al. [Wang et al. 2021] pursued this traditional method, but operated under the premise that collisions predominantly occur within confined areas. This assumption allows them to alleviate the complexity of the linear solving problem by transforming the system from a large, sparse matrix into a smaller, denser one.

Inspired by the above discussed previous works, we propose to use IPC for efficient collision handling in actuated face simulation. Furthermore, we notice that even a smaller number of the vertices in the confined region are involved in the collision, providing further room for optimization beyond the idea of Wang et al. [2021]. To this end, we are able to perform high-fidelity intersection-free simulation with optimized efficiency.

2 SIMULATION PROCEDURE

2.1 Actuated Soft Body Simulation

Unlike passive objects, the motion of muscles is driven by internal activation signals. To describe this mechanism, we can formulate the desired deformation as a symmetric actuation matrix [Ichim et al. 2017; Klár et al. 2020; Yang et al. 2022]. For the simulation mesh discretized into finite elements and nodal points, this shape-targeting model provides the following elastic potential for each element (denoted by subscript i):

$$E_i(x) = \min_{R_i \in \mathrm{SO}(3)} \|F_i(x) - R_i A_i\|_F^2 = \|G_i x - p_i\|^2, \tag{1}$$

where $x \in \mathbb{R}^{3n}$ is the stacked vector of the nodal vertices. The energy measures the distance between the deformation gradient F_i and the actuation matrix A_i using Frobenius norm. R_i is allowed to factorize out their rotational difference. Equivalently, the flattened deformation gradient is written as $G_i x$ where G_i is a gradient mapping matrix, and we use p_i to denote the flattened vector of $R_i A_i$. In the left part of Figure 1, the simulation mesh is colored based on the strength of actuation. On the other hand, the surface mesh of the face (the transparent layer) is *embedded* inside the simulation mesh via interpolation and deformed accordingly.

The quasi-static simulation aims to find the equilibrium state with minimal energy. This optimization problem could be effectively solved using Projective Dynamics (PD) [Bouaziz et al. 2014] in a local-global alternating manner. In the *local step*, we find the best rotation R_i using the polar decomposition of $F_i A_i$. This step could be parallelized among finite elements.

Then in the *global step*, $\{R_i\}$ are fixed. The total energy $E(x) = \sum_i w_i E_i(x)$ is simply the sum of all element energies, weighted by the volume w_i at rest state. By setting $\nabla E(x) = 0$ we get:

$$\underbrace{(\sum_i w_i G_i^\mathsf{T} G_i)}_{H} x = \sum_i w_i G_i^\mathsf{T} p_i. \tag{2}$$

The efficiency of PD comes from the fact that the left side H in the global step only depends on the configuration of the FEM system, which remains constant regardless of the actuation state. Thus, its Cholesky factorization could be pre-computed. At runtime, Equation 2 could be effectively solved by forward/backward substitution.

2.2 Collision Handling

The actuation by itself could drive the face into implausible states with self-intersection. To achieve visually convincing results, we adopt the method of Incremental Potential Contact (IPC) [Li et al. 2020] to handle collisions. For each primitive pair (vertex-triangle or edge-edge), their unsigned distance d_k is used to compute a barrier function $b(d_k)$ that penalizes penetration. $b(\cdot)$ is logarithm-like so that the response goes to infinity as the distance approaches zero, similar to the idea of the interior point method. The barrier is activated only when the distance is smaller than a distance threshold d_0, thus this threshold determines a constraint set C that contains the active primitive pairs.

Now we can append the energy of $E(x)$ with all the barriers:

$$B(x) = \kappa \sum_{k \in C} b(d_k(x)), \tag{3}$$

where κ is a stiffness parameter. Then we are simply facing another unconstrained optimization target $\hat{E}(x) = E(x) + B(x)$. For convenience, we define $\mathcal{B}(x) \triangleq \nabla^2 B(x)$, and the new global step is:

$$\underbrace{(H + \mathcal{B}(x))}_{\hat{H}(x)} \delta x = -\underbrace{(\sum_i w_i G_i^\mathsf{T} (G_i x - p_i) + \nabla B(x))}_{\hat{g}(x)}. \tag{4}$$

In IPC, continuous collision detection (CCD) is used to find the largest possible step size that does not cause penetration. Starting from the maximal step size, back-traced line search is performed to ensure convergence. We list the complete procedure of the solver in Algorithm 1.

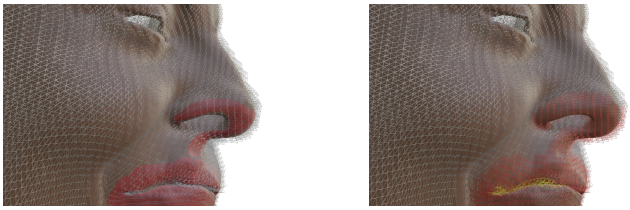
Algorithm 1 PD Solver with IPC Contact

Input x_t ▷ Equilibrium state at frame t (initial guess)
Output x_{t+1} ▷ Equilibrium state at frame $t+1$

$x \leftarrow x_t$
while not converged **do**
  for all i **do**
    $p_i \leftarrow \mathrm{GetProjection}(x, i)$ ▷ Equation 1
  end for
  $C \leftarrow \mathrm{UpdateConstraintSet}(x, d_0)$
  $\hat{H} \leftarrow H + \mathrm{PositiveDefinite}(\mathcal{B}(x, C))$
  $\hat{g} \leftarrow \sum_i w_i G_i^\mathsf{T}(G_i x - p_i) + \nabla B(x)$
  $\delta x \leftarrow -\hat{H}^{-1}\hat{g}$ ▷ Equation 4, Section 3
  $\alpha_m \leftarrow \mathrm{CCD}(x, dx)$
  $\alpha \leftarrow \mathrm{LineSearch}(x_t, \delta x, \alpha_m)$
  $x \leftarrow x_t + \alpha \cdot \delta x$
end while
$x_{t+1} \leftarrow x$

3 OPTIMIZING THE GLOBAL STEP

The fact that $\hat{H}(x)$ in Equation 4 is no longer constant prevents us from utilizing Cholesky pre-factorization for PD. Re-factorizing the system per iteration is apparently not efficient. Also note that $\hat{H}$ is sparse: given a vertex, its corresponding entries in the Hessian could only be non-zero if another vertex is in the same element (for H) or active collision primitive pair (for $\mathcal{B}$). Thus, an iterative method such as conjugate gradient (CG) is a reasonable choice.



(a) (b)

Figure 2: (a) We use the underlying embedded surface mesh of the face as the proxy for collision response. Specifically, the collision proxy is mainly the mouth region (red). (b) The simulation mesh could be divided into the collision-agnostic part (gray, n_1 vertices) and the collision-aware part (red, n_2 vertices) that embeds the proxy. During simulation, only a small fraction of the proxy vertices are activated in the constraint set C and affects the simulation mesh (yellow, n_c vertices). Note how the magnitude of these three hierarchies differ.

Furthermore, the constant factors $LL^\mathsf{T} = H$ is able to approximate $\hat{H}$ for pre-conditioning the system:

$$(L^{-1}\hat{H}(x)L^{-\mathsf{T}})L^\mathsf{T}\delta x = -L^{-1}\hat{g}(x). \tag{5}$$

We use the pre-conditioned CG solver as a baseline method to benchmark our subsequent optimizations.

3.1 Localizing Collision via Schur Complement

Our first consideration is that we do not want to solve a sparse, large system $\hat{H}$. Inspired by Wang et al. [Wang et al. 2021], we transform this problem into solving a dense, local matrix. Their implementation does not use IPC, but the idea suits our project. The only prerequisite is that we need to manually mark the elements (and their attached vertices) for which we would like to handle collision. This is a reasonable simplification given that collision predominantly happens in specific regions of the face during animation.

In practice, the simulation mesh x is coarse and does not represent the geometry of the face precisely, which makes it unsuitable for collision detection (see Figure 2a). Therefore, we instead employ the embedded surface s, which is linearly interpolated as $s = Wx$. Then s serves as the proxy for calculating the primitive pair distance, barrier energy and other quantities in IPC. In this way, we need to map the gradient and Hessian of barriers back to the original space of x, which is given by $\nabla B = W^\mathsf{T}\nabla B(s)$ and $\mathcal{B} = W^\mathsf{T}\nabla^2 B(s)W$.

Since s is only required for the prescribed region, we could split x into two folds: $x_1 \in \mathbb{R}^{3n_1}$ that are free of collision, and $x_2 \in \mathbb{R}^{3n_2}$ that embeds s. The idea is shown in Figure 2b. We permute the system $\hat{H}$ to cluster these two kinds of vertices:

$$P\hat{H}P^\mathsf{T} = P(H+\mathcal{B})P^\mathsf{T} = \begin{pmatrix} H_{11} & H_{12} \\ H_{21} & \hat{H}_{22} \end{pmatrix}, \tag{6}$$

where P is a permutation matrix. Multiplying P and P^T permutes rows and columns respectively. After permutation, $\mathcal{B}$ is only non-zero at the bottom right block that corresponds to x_2, thus only $\hat{H}_{22} = H_{22} + \mathcal{B}_{22}$ is non-constant. We pre-factorize the collision-free block $H_{11} = L_1 L_1^\mathsf{T}$ before simulation. Now the system could be decomposed as follows, which is an intermediate result of the incomplete Gaussian elimination:

$$\begin{pmatrix} H_{11} & H_{12} \\ H_{21} & \hat{H}_{22} \end{pmatrix} = \begin{pmatrix} L_1 & 0 \\ H_{21}L_1^{-\mathsf{T}} & I \end{pmatrix} \begin{pmatrix} I & 0 \\ 0 & \hat{\Sigma} \end{pmatrix} \begin{pmatrix} L_1^\mathsf{T} & L_1^{-1}H_{12} \\ 0 & I \end{pmatrix}, \tag{7}$$

where $\hat{\Sigma}$ is known as the Schur complement of H_{11}:

$$\hat{\Sigma} = \hat{H}_{22} - H_{21}H_{11}^{-1}H_{12} = \mathcal{B}_{22} + \underbrace{H_{22} - H_{21}H_{11}^{-1}H_{12}}_{\Sigma}. \tag{8}$$

The significance of this transformation is that we have restricted the influence of IPC to a local matrix $\hat{\Sigma}$. The system in Equation 7 can now be solved in 3 steps, simply by solving a lower triangular, a diagonal and an upper triangular system subsequently. Thereafter, the only heavy part is to solve a small, dense system $\hat{\Sigma}^{-1}$ of size $3n_2$, instead of the original sparse $\hat{H}$ of $3n$.

3.2 Low-Rank Inverse Update

The previous optimization strategy relies on our manual configuration, which does not consider the actual situation of the collision during the simulation. In practice, it is likely that not all the vertices of x_2 are affected by IPC (see Figure 2b). Therefore, we further consider how to accelerate solving $\hat{\Sigma}^{-1}$ if only a small fraction of entries has changed during the runtime. Again, we apply the same permutation trick on $\mathcal{B}_{22}$, this time with a matrix $Q \in \mathbb{R}^{3n_2 \times 3n_2}$ that picks out its non-zero entries $\check{\mathcal{B}} \in \mathbb{R}^{3n_c \times 3n_c}$:

$$Q\mathcal{B}_{22}Q^\mathsf{T} = \begin{pmatrix} \check{\mathcal{B}} & 0 \\ 0 & 0 \end{pmatrix} \Leftrightarrow \mathcal{B}_{22} = Q^\mathsf{T}\begin{pmatrix} \check{\mathcal{B}} & 0 \\ 0 & 0 \end{pmatrix} Q. \tag{9}$$

Since the permutation of the remaining zeros is arbitrary, this representation can be simplified as

$$\mathcal{B}_{22} = \begin{pmatrix} \check{Q}^\mathsf{T} & \cdots \end{pmatrix} \begin{pmatrix} \check{\mathcal{B}} & 0 \\ 0 & 0 \end{pmatrix} \begin{pmatrix} \check{Q} \\ \vdots \end{pmatrix} = \check{Q}^\mathsf{T}\check{\mathcal{B}}\check{Q}, \tag{10}$$

where $\check{Q} \in \mathbb{R}^{3n_c \times 3n_2}$ and n_c is the number of vertices affected by the active barriers.

We pre-compute the explicit inversion of the Schur complement Σ^{-1} from the PD part as a dense matrix. Thereafter, $\hat{\Sigma}^{-1}$ can be updated efficiently according to the following low-rank update formula (known as Woodbury identity [Hager 1989]):

$$\begin{aligned} \hat{\Sigma}^{-1} &= (\Sigma + \check{Q}^\mathsf{T}\check{\mathcal{B}}\check{Q})^{-1} \\ &= \Sigma^{-1} - \Sigma^{-1}\check{Q}^\mathsf{T}(\check{\mathcal{B}}^{-1} + \check{Q}\Sigma^{-1}\check{Q}^\mathsf{T})^{-1}\check{Q}\Sigma^{-1}. \end{aligned} \tag{11}$$

For each iteration, we need to invert two matrices, namely $\check{\mathcal{B}}^{-1}$ and $(\check{\mathcal{B}}^{-1} + \check{Q}\Sigma^{-1}\check{Q}^\mathsf{T})^{-1}$. Once again we have shrunk the scale of the problem from $3n_2$ to $3n_c$. The drawback of this approach is that we need to explicitly invert matrices now, while the previous optimization only requires solving linear systems. Nevertheless, this optimization strategy turns out to be effective because n_c is usually smaller than n_2 by magnitudes.

4 EXPERIMENT

Our solver is written in Python and runs on GPU (CUDA). Through Python wrapper libraries we employ cuSPARSE for sparse triangular solving and cuSOLVER for dense matrix inversion. Cholesky factorization is performed using CHOLMOD [Chen et al. 2008]. As for IPC, we integrate the open-source implementation from the original authors [Ferguson et al. 2020] which only runs on CPU.

We firstly show the effect of collision handling in Figure 3, where we compare three methods: no collision handling; the spring-like

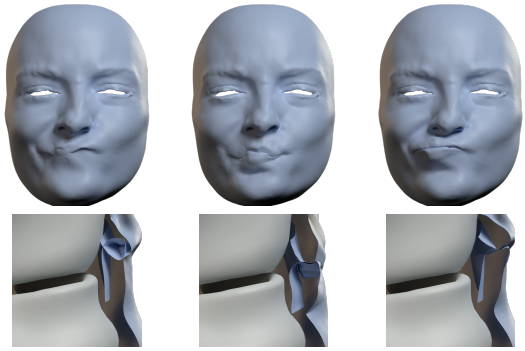


Figure 3: The effect of collision handling. The actuation may lead to severe self-intersection artifacts in extreme cases. Both IPC (ours) and the standard PD can solve the penetration, but IPC results in more natural collision responses.

repulsion method in the standard PD; ours using IPC. Compared to the collision solution in the standard PD, IPC guarantees the simulation is intersection-free and separates the colliding parts more naturally, thanks to its repulsive contact barrier.

To benchmark the performance of our optimization for collision, we test our solver on a dataset of three facial animation sequences, with each containing 50 frames. We run the experiment on a PC equipped with an Intel i9-10900KF CPU and a NVIDIA RTX 3060 GPU. The detailed example setup and performance information is listed in Table 1. Note that the prescribed colliding vertices n_2 is one magnitude smaller than n_1, and during simulation the actual n_c is typically again no more than 10% of n_2, which explains why the optimization works.

Compared to the baseline using conjugate gradient solver, our final optimized version achieves around 3x speedup for the final frame time and 5-7x speedup for the global step. Both optimization strategies roughly contribute equally to the speedup. Moreover, our optimized algorithm has another advantage over the iterative solver. The pre-conditioner in Equation 5 becomes less accurate to approximate $\hat{H}^{-1}$ if the collision stiffness κ becomes larger. In IPC, κ is initialized according to the average stiffness of the soft body and adaptively adjusted. If we increase its initial value by 10x, the cost of the conjugate gradient solver would nearly double. However, our method is a direct solver, which does not rely on a pre-conditioner and performs independent of κ.

Collision handling using IPC is still a relatively expensive solution, and even our optimized version still requires 4x frame time compared to the PD without collision. Besides the fact that the system cannot be easily pre-factorized, the accompanied cost of Hessian computation, CCD and line search with IPC is also significant. In our final optimized implementation, they almost occupy half of the total frame time and would hinder the overall benefit for further optimizing the global step only.

5 CONCLUSIONS

Our solver is able to animate human face and is aware of collision. Currently, our optimization for collision handling still relies on manually prescribing the region of interest, and it would be beneficial to explore fully automatic solutions. Moreover, exploiting collision detection on GPU and faster CCD approximation may help address our performance bottleneck caused by IPC. Also, the idea of our optimization might find application in other problems with a similar setting, which involves solving a system with a fixed and varying part. Finally, though our work is implemented for quasi-static actuated face simulation, it should be straightforward to extend it to other types of soft bodies and dynamics simulation.

Table 1: The average per-frame performance data for three test cases. The meaning of the steps are: IPC: constructing the IPC constraint set, computing the gradient and Hessian (with positive-definite regularization) of the barrier function; G-Step: solving the global step; CCD: continuous collision detection; LS: line search; Misc: the remaining items, including the local step, matrix assembly, data transfer between CPU/GPU, etc. The time is measured in milliseconds. For each test case, we experiment with four configurations: ○: without collision; △: IPC with pre-conditioned conjugate gradient solver; □: optimization as in Section 3.1; ◇: additional optimization as in Section 3.2.

	n_1	n_2	IPC	G-Step	CCD	LS	Misc	Total	
Case 1	51.5k	3.7k	—	418	—	—	173	591	○
			451	6059	348	361	171	7390	△
			472	2923	401	378	172	4346	□
			470	**1400**	396	355	166	**2787**	◇
Case 2	53.0k	3.9k	—	407	—	—	182	589	○
			412	6861	448	493	228	8442	△
			428	2978	463	405	154	4427	□
			426	**976**	416	399	158	**2375**	◇
Case 3	59.0k	4.1k	—	436	—	—	191	627	○
			419	6101	274	290	156	7240	△
			434	3163	275	334	177	4383	□
			436	**1013**	274	325	170	**2217**	◇

REFERENCES

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, Article 154 (jul 2014), 11 pages. https://doi.org/10.1145/2601097.2601116

Yanqing Chen, Timothy A. Davis, William W. Hager, and Sivasankaran Rajamanickam. 2008. Algorithm 887: CHOLMOD, Supernodal Sparse Cholesky Factorization and Update/Downdate. *ACM Trans. Math. Softw.* 35, 3, Article 22 (oct 2008), 14 pages. https://doi.org/10.1145/1391989.1391995

Zachary Ferguson et al. 2020. *IPC Toolkit.* https://ipc-sim.github.io/ipc-toolkit/

W. W. Hager. 1989. Updating the Inverse of a Matrix. *SIAM Rev.* 31, 2 (jun 1989), 221–239. https://doi.org/10.1137/1031049

Alexandru-Eugen Ichim, Petr Kadleček, Ladislav Kavan, and Mark Pauly. 2017. Phace: Physics-Based Face Modeling and Animation. *ACM Trans. Graph.* 36, 4, Article 153 (jul 2017), 14 pages. https://doi.org/10.1145/3072959.3073664

Gergely Klár, Andrew Moffat, Ken Museth, and Eftychios Sifakis. 2020. Shape Targeting: A Versatile Active Elasticity Constitutive Model. In *ACM SIGGRAPH 2020 Talks* (Virtual Event, USA) *(SIGGRAPH '20)*. Association for Computing Machinery, New York, NY, USA, Article 59, 2 pages. https://doi.org/10.1145/3388767.3407379

Lei Lan, Guanqun Ma, Yin Yang, Changxi Zheng, Minchen Li, and Chenfanfu Jiang. 2022. Penetration-Free Projective Dynamics on the GPU. *ACM Trans. Graph.* 41, 4, Article 69 (jul 2022), 16 pages. https://doi.org/10.1145/3528223.3530069

Minchen Li, Zachary Ferguson, Teseo Schneider, Timothy Langlois, Denis Zorin, Daniele Panozzo, Chenfanfu Jiang, and Danny M. Kaufman. 2020. Incremental Potential Contact: Intersection-and Inversion-Free, Large-Deformation Dynamics. *ACM Trans. Graph.* 39, 4, Article 49 (aug 2020), 20 pages. https://doi.org/10.1145/3386569.3392425

Qisi Wang, Yutian Tao, Eric Brandt, Court Cutting, and Eftychios Sifakis. 2021. Optimized Processing of Localized Collisions in Projective Dynamics. *Computer Graphics Forum* 40, 6 (2021), 382–393. https://doi.org/10.1111/cgf.14385 arXiv:https://onlinelibrary.wiley.com/doi/pdf/10.1111/cgf.14385

Lingchen Yang, Byungsoo Kim, Gaspard Zoss, Baran Gözcü, Markus Gross, and Barbara Solenthaler. 2022. Implicit Neural Representation for Physics-Driven Actuated Soft Bodies. *ACM Trans. Graph.* 41, 4, Article 122 (jul 2022), 10 pages. https://doi.org/10.1145/3528223.3530156