Eurographics Symposium on Parallel Graphics and Visualization (2019)
H. Childs, S. Frey (Editors)

Parallel XPBD Simulation of Modified Morse Potential - an Alternative Spring Model

Ozan Cetinaslan †

Instituto de Telecomunicações & Faculdade de Ciências, Universidade do Porto, Portugal

Abstract

In this paper, we introduce a modified Morse potential as an alternative to the existing spring models within a massively parallel extended Position Based Dynamics (XPBD) algorithm. To date, stretching is one of the most popular constraint types of XPBD frameworks due to its simplicity, robustness and efficiency. However, the underneath mathematical expression of stretching constraint does not fully represent a spring model and behaves too stiff over a certain iteration count or damping coefficient. On the other hand, Hookean spring potential behaves softer and viscoelastic within the XPBD algorithm under the same conditions as stretching constraint. Our modified Morse potential addresses this issue by keeping the simulation of deformable models in between Hooke's law and stretching constraint. To demonstrate the benefits of modified Morse potential with higher frame rates, we develop an efficient Independent Edge Grouping algorithm for XPBD method which provides parallel processing on GPU. We compare the simulation results of cloth and volumetric models with stretching constraint, Hookean and St. Venant-Kirchhoff (STVK) spring potentials. We believe that our modified Morse potential is easy to implement and seamlessly fit into the existing XPBD frameworks.

CCS Concepts

• ***Computing methodologies*** → *Physical simulation;*

1. Introduction

Position Based Dynamics (PBD) [MHHR07] has been a widely used method in games, movies and medical applications. This can be explained with its simplicity, stability and performance. PBD formulates the physical phenomena based on the projected position constraints and the velocities are computed by using the iterated positions. Due to the versatile aspect of PBD, large spectrum of position constraints were explored to simulate geometrically motivated constraints, finite element models, rigid bodies or fluids. Recently, [MMC16] improved the existing PBD algorithm (XPBD) by applying a total Lagrange multiplier for general material stiffness. In this paper, we use the XPBD technique as our general simulation algorithm.

The stretching constraint is the most basic projected position constraint of PBD. It simplifies the spring potential within the Gauss-Seidel (GS) iteration portion of the algorithm. The approximation of stretching constraint works fine up to a certain number of GS iterations. However, it responds too stiff under the high GS iteration count and dissipate energy quicker than expected under relatively higher damping coefficients. Due to visually unpleasant results, this fact is not desired for some arbitrary simulation cases, such as cloth simulation. On the other hand, [BKCW14] already proved that the PBD algorithm is capable of simulating potential energy density functions of hyperelastic materials. Inspired by the approach of [BKCW14], we applied the Hookean and St. Venant-Kirchhoff (STVK) spring potentials within the XPBD algorithm. Under the exact same conditions of stretching constraint, Hooke's law produces relatively softer visual results and STVK spring potential from [RLK18] behaves almost similar to stretching constraint.

In this paper, we modify and adapt Morse potential [Mor29] for XPBD algorithm as an alternative spring model for 2D and 3D deformable object simulations. Morse potential defines the potential energy of the diatomic molecules in the field of molecular mechanics. However, direct application of Morse potential is not feasible to produce visually pleasant results like the commonly used spring potentials. Therefore, we modified the original formulation of Morse potential and applied it within the XPBD algorithm as an alternative spring constraint. We observe that it has a great potential to produce interactive physics simulations along with the other spring potentials, such as Hooke's law and STVK spring. Moreover, we analyze the advantages and disadvantages of spring potentials and widely used stretching constraint under collision and damping. Besides, we couple them with the volume constraint to simulate 3D volumetric models and study their behaviors. We believe that our modified Morse potential is easy to implement similar to the previ-

† ozan.cetinaslan@gmail.com

DOI: 10.2312/pgv.20191108

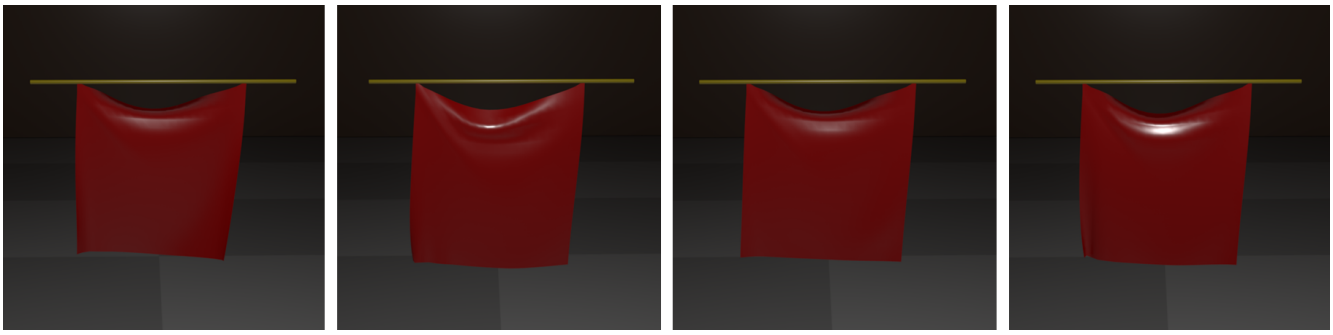

Figure 1: *A piece of cloth (with 4880 edges) hangs under constant gravitational force. From left to right: Stretching Const., Hookean Spring Const., STVK Spring Const., Our Modified Morse Pot. Const. All simulations are executed under the same conditions, such as* $\alpha = 0.00001$*, iteration count* $= 200$*, damping coef. varies with* $0.0, 0.3$ *and* 1.0*, step-size* $= 1/24$*, spring stiffness* $= 10.0$*.*

ous potentials, has some advantages under some circumstances and is a new flavor for mass-spring simulations.

XPBD method is known by its unique type of Gauss-Seidel solver. It provides unconditional stability by the fact that projected constraints are solved iteratively one after another and this cycle repeats itself for a certain number of iterations. Besides, XPBD defines the simulated model as a particle system such that each particle is connected with each other by predefined constraints. In this paper, edge and particle couple represents the mass spring system. Therefore, the strict connectivity of the model limits us to directly apply the parallel processing to position based algorithms. For that reason, we develop an efficient and straightforward algorithm, "Independent Edge Grouping", to overcome this problem. Our algorithm is an intuitive and simplified version of graph coloring method and provides massive parallelization on GPU.

The rest of the paper is organized as follows: section 2 describes the related work for position-based dynamics techniques and spring potentials. Section 3 presents the basic background and foundations. Section 4 describes our modified Morse potential in detail. Section 5 describes some additional required constraints. Section 6 presents our approach to parallelize the XPBD method. Section 7 demonstrates the results and compares with the existing potentials. Section 8 concludes the paper.

2. Related Work

Mass-spring networks are highly employed models to simulate elastic materials due to their straightforward theory and implementation. In practice, the inner forces are derived from the potential energy of the spring. Hooke's law (from 1660) is the popular spring potential for simulating dynamic systems. Early works (Terzopoulos et al. [TPBF87], Breen et al. [BHW94] and Baraff and Witkin [BW98]) employed Hookean spring potential with implicit or semi-implicit integration techniques for cloth and deformable objects simulations. More recently, Lui et al. [LBOK13] presented a local/global approach for the Hookean spring potential by expressing the implicit Euler integration as an energy minimization problem. In terms of applications, mass-spring systems are widely used to simulate one, two and three-dimensional objects such as hair (Selle et al. [SLF08]), cloth (Bridson et al. [BMF03]) and volumetric elastic solids (Gripsun et al. [GHDS03]). A detailed information on classical mass-spring based simulations can be found in the survey of Nealen et al. [NMK*06].

On the other hand, Position Based Dynamics (PBD) is a successful technique as an alternative to force and velocity based algorithms. Early foundations of PBD can be found in [GG94] for rigid body simulation and in [Fau99] for deformable models. After a short time, Jakobsen [Jak01] presented the technique of rigid and deformable body simulations in the first version of Hitman: Codename 47 game engine. He described the Verlet integration with the direct position manipulation aspect. Müller et. al. [MHHR07] generalized the idea of [Jak01] with corrected velocities, and introduced the formal PBD algorithm. PBD defines the potential energies as projected constraints. Goldenthal et al. [GHF*07] applied the constraints for their fast projection algorithm. Besides, Autodesk Maya employs a unified constrained based solver which is called Nucleus. Stam [Sta09] presented the details of Nucleus which works similar to PBD. More recently, Tournier et al. [TNGF15] applied second order derivatives on the constraints in order to avoid the instabilities. Recently, Macklin et al. [MMC16] extended the PBD algorithm (XPBD) with a total Lagrange multiplier to address the iteration count and simulation step-size dependencies. This extension is inspired by [SLM06]. In this paper, we use the same base algorithm as XPBD. For more detailed information on PBD, we refer the reader to [BMM17].

Due to simplicity and efficiency, many researchers have taken advantage of PBD to simulate solid, fluid, rigid body dynamics, cloth and skin deformations. Müller and Chentanez addressed the details for cloth and skin deformations with wrinkle meshes [MC10], and later on they introduced oriented particles [MC11] for rigid, elastic and plastic deformations. Kim et al. [KCMF12] simulated inextensible cloths with long range attachments. PBD approach is also advantageous to simulate hair, fur [MKC12] and fluid dynamics [MM13] as well. Furthermore, Müller et al. [MCKM14] and Bender et al. [BKCW14] proved that finite element based hyperelastic materials can be simulated by PBD. Moreover, it is possible to see PBD algorithm in facial animation frameworks. Fratarcangeli [Mar12] defined a muscle and skinning system for facial

Figure 2: *A flag (with 4880 edges) blows under constant wind force. From left to right: Stretching Const., Hookean Spring Const., STVK Spring Const., Our Modified Morse Pot. Const. All simulations are executed under the same conditions, such as* $\alpha = 0.00001$*, iteration count* $= 100$*, damping coef.* $= 1.0$*, step-size* $= 1/24$*, spring stiffness* $= 10.0$*.*

models that incorporates with the PBD method. Cetinaslan and Orválho [CO16] proposed a framework for local contact deformations on facial models that has employed the constrained dynamics. Elastic rod simulations have been taking the advantage of the PBD method. Umetani et al. [USS14] and Deul et al. [DKWB18] proposed novel PBD based algorithms to simulate complex behavior of elastic rods.

There are some novel algorithms that have addressed the performance of PBD. Müller [MÖ8] presented a multi-grid technique to improve the performance of the solver. Besides, two notable techniques were introduced recently. One is the Chebyshev approach by Wang [Wan15] that accelerates the iteration process during cloth and solid simulations. The other is the graph coloring method by Fratarcangeli et al. [FTP16] that heavily parallelizes the PBD iterations. Both approaches are quite popular due to their significant performance enhancements. In this paper, we have utilized a simplified version of graph coloring algorithm which only addresses to group the non-adjacent edges of the mesh similar to [FP14].

In this paper, we have studied and taken advantage of the previous works. We observe that the stretching constraint of PBD technique has been used over the years without any alternative. Although, Strain Based Dynamics [MCKM14] can be considered as an alternative to stretching and shearing constraints, it requires more effort for implementation. Due to the molecular mechanics roots of XPBD method, we modify the original Morse potential from [Mor29] and adapt it to XPBD algorithm as a novel spring constraint. The results show that our modified Morse potential can be considered as a new alternative spring model for the mass-spring simulations.

3. Background

PBD [MHHR07] and XPBD [MMC16] accept the input model as a system of interconnected particles with a given set of position constraints. Both algorithms work in three steps: As a first step, the predicted positions of each particle ($x_i \in \mathbb{R}^3$) is computed by employing an integration scheme. In the second step, the bilateral constraint set ($C(x) = 0$) is solved by using an iterative Guass-Seidel approach in order to locate the particles to their final positions. Lastly, the velocities are updated according to the new particle positions. The second step can be considered as the core of the algorithm. The major goal is to determine the position correction of each particle (Δx_i) such that bilateral behavior is conserved ($C(x+\Delta x) = 0$). During this step, linear and angular momentum are conserved implicitly, and the displacements of the vertices are calculated by employing the Taylor-expansion ($C(x+\Delta x) \approx C(x) + \nabla_x C(x) \cdot \Delta x = 0$). Therefore, Δx is computed as in Equation (1):

$$\Delta x_i = w_i \nabla_{x_i} C(x) \lambda_i \tag{1}$$

where w_i is the inverse particle mass ($1/m$) and λ_i stands for the Lagrange multiplier (in Equation (2)) which is obtained by substituting Equation (1) into the aforementioned Taylor-expansion:

$$\lambda_i = -\frac{C(x)}{\sum_i w_i |\nabla_{x_i} C(x)|^2} \tag{2}$$

XPBD [MMC16] alters the original PBD formulation with total Lagrange multiplier in order to bring a solution to stiffness dependency on simulation step-size and iteration count. In XPBD, λ_i from Equation (1) is replaced with $\Delta\lambda_i$. Moreover, $\Delta\lambda_i$ is extended with an inverse stiffness parameter, α. As a result, the new position update is computed as in Equation (3):

$$\Delta x_i = w_i \nabla_{x_i} C(x) \Delta\lambda_i \tag{3}$$

where $\Delta\lambda_i$ is redefined as in Equation (4):

$$\Delta\lambda_i = -\frac{C(x) + \tilde{\alpha}\lambda_i}{(\sum_i w_i |\nabla_{x_i} C(x)|^2) + \tilde{\alpha}} \tag{4}$$

Algorithm 1 XPBD Algorithm

loop
  $x_{n+1} = x_n + hv_n + h^2 w_i f_{ext}$
  initialize total Lagrange multiplier $\lambda_0 = 0$
  while `k < iterationCount` **do**
    for `each Constraint` **do**
      compute $\Delta\lambda$ (Eq. (4))
      compute Δx (Eq. (3))
      update $\lambda_{k+1} = \lambda_k + \Delta\lambda$
      update $x_{k+1} = x_k + \Delta x$
    end for
    $k = k + 1$
  end while
  update positions $x_{n+1} = x_k$
  update velocities $v_{n+1} = (x_{n+1} - x_n)/h$
end loop

where $\tilde{\alpha} = \alpha/h^2$, $\lambda_{i+1} = \lambda_i + \Delta\lambda_i$ that is computed in each iteration, and h represents the simulation step-size. It should be noted that the value of λ_i is updated incrementally for each constraint type at the instant iteration. The XPBD process is summarized in Algorithm 1.

4. Modified Morse Potential

The main idea of our approach is to reformulate the Morse potential energy function in a way that allows us to use it as a spring potential. Morse potential describes the energy and vibration between two atoms in molecular scale. In its simplest form [DS88], Morse potential is defined as in Equation (5):

$$E_{morse}(x) = D(1 - e^{-a(x-l_0)})^2 \tag{5}$$

where D and a are energy constants, l_0 is the rest length and x is the norm of the interatomic distance ($||x_2 - x_1||$). Furthermore, Equation (5) is not the only representation of the Morse potential. According to its application domain, Morse potential can be redefined by adding and subtracting constants. Therefore, it is possible to represent Morse potential in a different form ([Mor29] and [GW59]) by simply subtracting the constant D from equation (5). The new form is defined as in Equation (6):

$$E_{morse}(x) = De^{-2a(x-l_0)} - 2De^{-a(x-l_0)} \tag{6}$$

On the other hand, directly applying equation (5) or (6) is not feasible due to the fact that the constants D and a contain such small values from the atomic space. However, there exists an obvious relation between the stretching constraint and the exponential part of Equation (5). Therefore, we simplify the Morse potential from Eq. (5) and transform it into the known mass-spring potential base. Our modified Morse potential as a bilateral spring constraint is defined as in Equation (7):

$$C_{morse}(x_1, x_2) = k(1 - e^{(||x_2 - x_1|| - l_0)})^2 \tag{7}$$

where $k \geq 0$ is the spring stiffness, $x_1, x_2 \in \mathbb{R}^3$ are the spring endpoints and l_0 is the rest length. Our modification maintains the desired deformations in edge directions and takes advantage of the exponential part that produces continuous wrinkles in 2D cloth simulation cases and loads the constraint for plausible visual results in the simulation of 3D volumetric models.

The gradients of our modified Morse potential over x_1 and x_2 are actually associated with the inner forces of the spring and necessary for the position updates (Δx and $\Delta\lambda$) within the Gauss-Seidel iteration of Algorithm 1. The corresponding gradients are obtained as follows:

$$\nabla_{x_1} C_{morse}(x_1, x_2) = -2k \frac{e^{||x_2 - x_1||}(e^{||x_2 - x_1||} - 1)}{||x_2 - x_1||}(x_2 - x_1) \tag{8}$$

$$\nabla_{x_2} C_{morse}(x_1, x_2) = 2k \frac{e^{||x_2 - x_1||}(e^{||x_2 - x_1||} - 1)}{||x_2 - x_1||}(x_2 - x_1) \tag{9}$$

In order to obtain the final position updates, $C_{morse}(x_1, x_2)$ and its gradients are plugged into equation (4) for obtaining the instant and total Lagrange multipliers. After, the position update is computed by employing equation (3) for each point.

5. Additional Constraints

We need to apply some additional constraints in order to preserve the volume of the 3D volumetric models, handle the collision and damping for overall aesthetic. We summarize these constraints as:

5.1. Volume Constraint

Bilateral volume constraint from [BMM17] is employed to conserve the overall volume of the 3D model. After obtaining the desired surface deformation from stretching or spring potential constraints, the attachment of volume constraint provides the recovery of the volumetric model. Although the application of volume constraint is common in tetrahedral meshes, we provide a straightforward solution to apply this constraint to the volumetric triangle meshes. Despite the fact the triangle does not have volume, we compute the model's center of mass (by assuming masses are uniform) for each iteration, and use that center point as the tip of each "ghost" tetrahedron. The mathematical expression of volume constraint is shown as follows:

$$C_{volume}(x_0, x_1, x_2, x_3) = \frac{1}{6}\left(\Big((x_1 - x_0) \times (x_2 - x_0)\Big) \cdot (x_3 - x_0)\right) - V_0 \tag{10}$$

where $x_1, x_2, x_3 \in \mathbb{R}^3$ are the triangle points, $x_0 \in \mathbb{R}^3$ is the center of mass and V_0 is the rest volume. The corresponding gradients are obtained as follows:

$$\nabla_{x_1} C_{volume} = \frac{1}{6}\Big((x_2 - x_0) \times (x_3 - x_0)\Big) \tag{11}$$

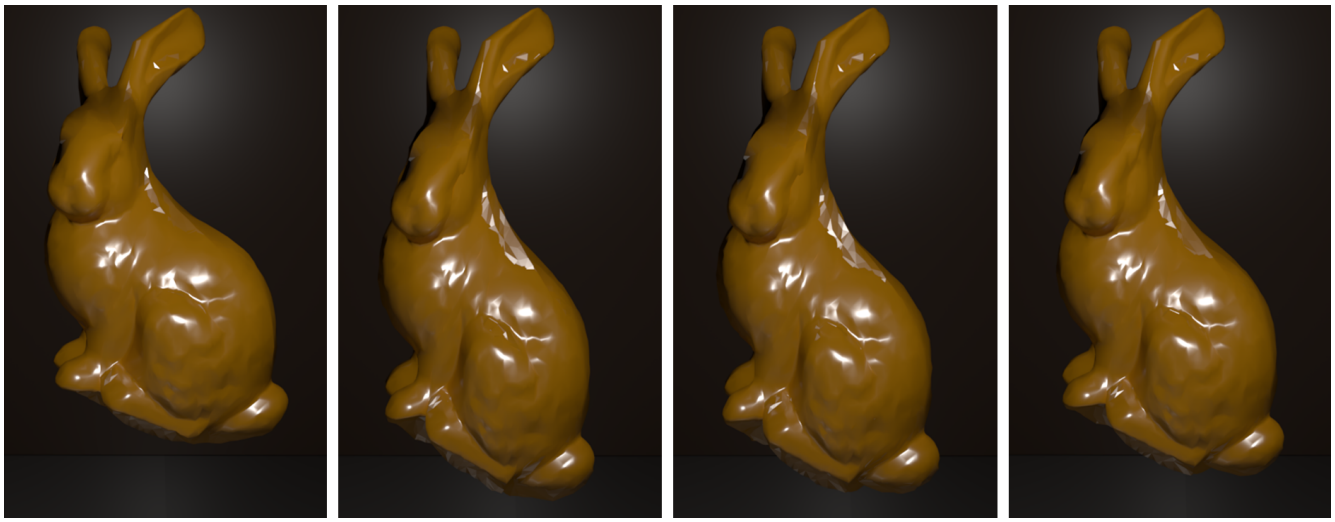

Figure 3: *Bunny model (with 12288 edges) falls under constant gravitational force. From left to right: Stretching Const., Hookean Spring Const., STVK Spring Const., Our Modified Morse Pot. Const. All spring constraints are coupled with the volume preservation constraint. All simulations are executed under the same conditions, such as* $\alpha = 0.0001$*, iteration count* $= 50$*, damping coef.* $= 0.0$*, step-size* $= 1/24$*, spring stiffness* $= 10.0$*.*

$$\nabla_{x_2} C_{volume} = \frac{1}{6}\Big((x_3 - x_0) \times (x_1 - x_0)\Big) \tag{12}$$

$$\nabla_{x_3} C_{volume} = \frac{1}{6}\Big((x_1 - x_0) \times (x_2 - x_0)\Big) \tag{13}$$

$$\nabla_{x_0} C_{volume} = -\nabla_{x_1} C_{volume} - \nabla_{x_2} C_{volume} - \nabla_{x_3} C_{volume} \tag{14}$$

By employing equations (4) and (3), we update the positions for the volume preservation except x_0 which is associated with the center of mass. We assign zero to its mass which indicates that the center of mass is locked with the infinite mass and does not have any effect in the final position updates.

5.2. Collision and Damping

Collision: In our framework, we employ two different unilateral collision constraints for sphere and convex object collisions. Sphere collision (Figure 5) is defined by $C(x) = R - |x - S_{cen}| \geq 0$ where R is the sphere radius and S_{cen} is the sphere center. For the collision of convex objects, we use the contact detection of the arbitrary surfaces. This is adapted from [BML*14]. We search the nearest collision point p with a surface normal n_s for the colliding particle x. Convex object collision (Figure 4) is satisfied by $C(x) = (x - p) \cdot n_s \geq 0$.

Damping: In dynamic simulations, damping is a significant factor to obtain the overall aesthetics. We use the damping model of [MMC16] in which Rayleigh dissipation function is employed: $R = \frac{1}{2}\dot{C}(x)^T \beta \dot{C}(x)$ where $C(x)$ is the constraint function and β is the diagonal matrix of damping coefficients. Due to the fact that the damping model is derived directly to the total Lagrange multiplier in XPBD [MMC16], it performs damping locally for each position correction within the Gauss-Seidel iteration. This provides a modest performance gain when compared to the damping of [MHHR07]. Nevertheless, extending the total Lagrange multiplier with the Rayleigh dissipation function makes the damping fully dependent on the compliance parameter (α). Therefore, assigning a lower value to the compliance parameter inherently decreases the damping. Therefore, we prefer keeping the value of α greater than 10^{-6}.

6. Parallel XPBD

Position Based Dynamics algorithm computes the projected constraints in a Gauss-Seidel type solver. Therefore, each constraint is solved sequentially from the first target primitive (in our case, the target primitives are the edges of the mesh which represent the conventional spring network) to the last one for every iteration. Al-

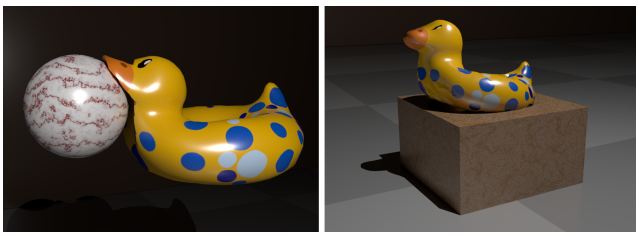

Figure 4: *Duck Lifebuoy model (with 9261 edges) is simulated with our modified Morse potential and volume constraints couple. Collision handling is tested. Left: Sphere collision constraint. Right: Convex objects collision constraint.*

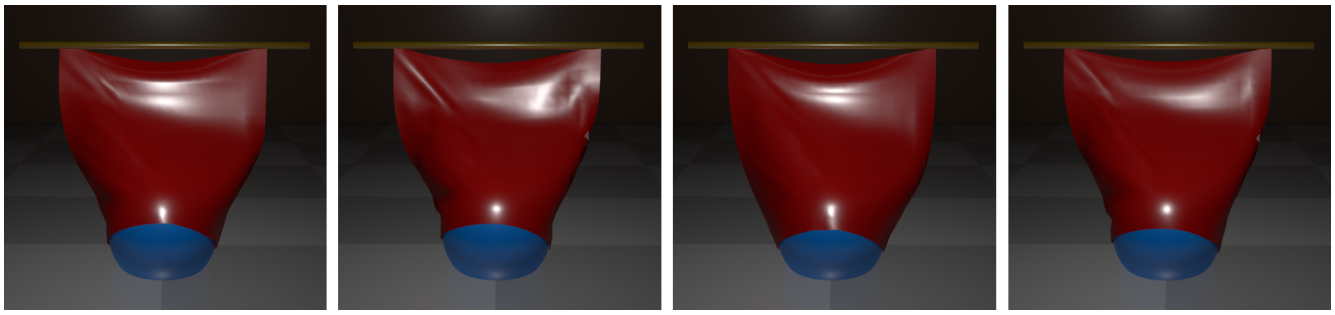

Figure 5: *A piece of cloth (with 4880 edges) falls under constant gravitational force and contacts with a static sphere object. From left to right: Stretching Const., Hookean Spring Const., STVK Spring Const., Our Modified Morse Pot. Const. All simulations are executed under the same conditions, such as* $\alpha = 0.00001$*, iteration count* $= 200$*, damping coef.* $= 0.3$*, step-size* $= 1/24$*, spring stiffness* $= 10.0$*.*

though high number of iterations increases the visual quality and precision, it causes a serious performance loss. GPU based parallelization may address this performance issue. However, the direct application of parallelization to the XPBD algorithm may lead to unpredicted results. For example, two primitives that share the same particle can be processed by two different threads at the same time. This fact ends up with the race condition and causes a violation to update the position of the particle.

To avoid this problem and take advantage of parallelization on GPU, we applied our own algorithm to split the mesh into nonadjacent edges groups. In each group, none of the members share the same particle. We have inspired from the Graph Coloring algorithm in [FP15] and [FTP16]. However, our algorithm is simplified and reduced to the edge space of the mesh. It is an iterative two phase algorithm that is summarized in Algorithm 2: In first phase, we create a list of arrays where each array contains the indices of all nonadjacent edges of each edge (lines 1-14 in Alg. 2). In the second phase, we compare each array members with each other according to their connectivity. If any two edges are connected to each other or exist in other independent edge groups, they are located to different groups (lines 15-37 in Alg. 2). Since our models are consist of triangular meshes, we noticed that 6 groups of edges are sufficient to hold all independent edges. This whole process is performed during the initialization phase. After, all independent edge groups are solved for XPBD simulation in parallel on the GPU. Figure 6 demonstrates our algorithm with a straight-forward triangular mesh example.

7. Implementation and Results

We have implemented our own version of parallel XPBD algorithm [MMC16] as a plugin for Autodesk Maya by using C++/CUDA that works on GPU. Each independent edge group member has been implemented in a different kernel function. Therefore, all kernels have been processed on GPU simultaneously. This allows us to utilize massive number threads during the computations. Besides, we have implemented a sequential version of XPBD on CPU by using C++ for comparison purpose. 2D and 3D triangle meshes have been used for the experiments. All test scenarios presented within this paper have been performed on a 4-core Intel i7-2600 3.4 GHz machine with 8 GB of RAM and an nVidia GTX 570 GPU.

Algorithm 2 Independent Edge Grouping Algorithm

```
1:  NumberofEdges = TotalNumberofEdges
2:  while EdgeID < NumberofEdges do
3:      AdjArray = GetConnectedEdges(EdgeID)
4:      for i = 0 to NumberofEdges do
5:          if i is EdgeID then
6:              continue
7:          else if i ∈ AdjArray then
8:              continue
9:          else
10:             NonAdjArray <-pushback- i
11:         end if
12:     end for
13:     NonAdjArraysList[edgeID] = NonAdjArray
14: end while
15: for all IndEdgesArray ∈ IndEdgesArraysList do
16:     if IndEdgesArray is IndEdgesArraysList[0] then
17:         IndEdgesArray <-pushback- 0
18:     end if
19:     SourceArray = NonAdjArray from the list
20:     for i = 0 to sizeofSourceArray do
21:         TargetIndex = SourceArray[i]
22:         TargetArray = NonAdjArraysList[TargetIndex]
23:         for j = 0 to sizeofSourceArray do
24:             if SourceArray[j] is TargetIndex then
25:                 continue
26:             else if SourceArray[j] ∈ TargetArray or
                SourceArray[j] ∈ ∀IndEdgesArray then
27:                 continue
28:             else
29:                 Erase SourceArray[j]
30:                 j = j - 1
31:             end if
32:         end for
33:     end for
        IndEdgesArray = SourceArray
34: end for
```

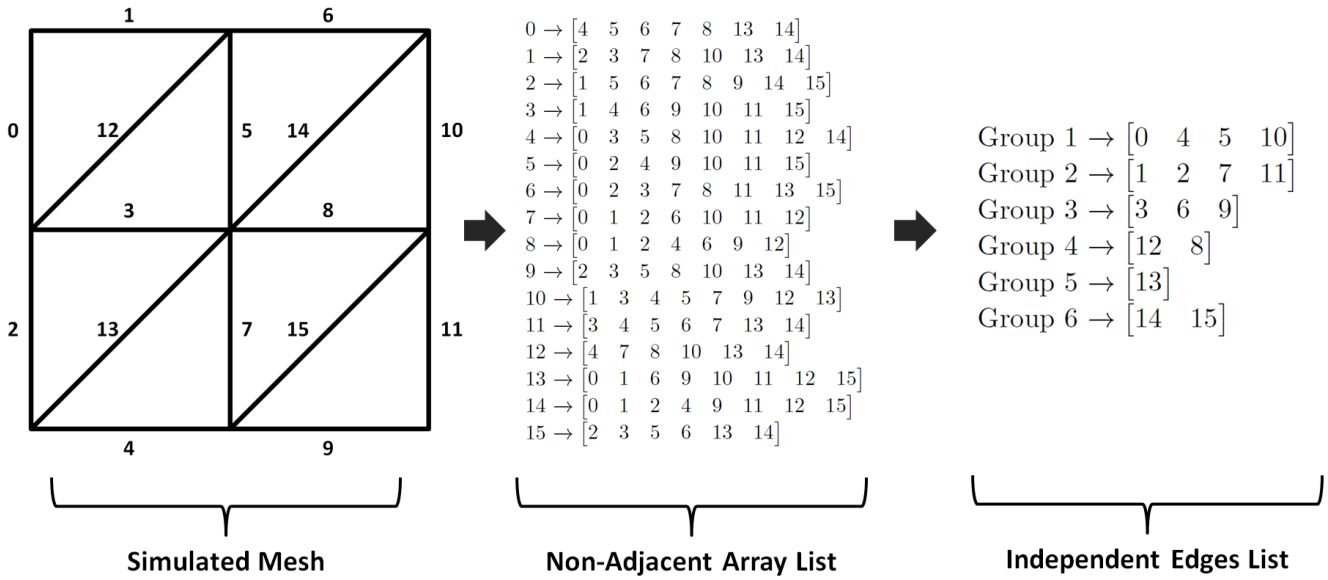

Figure 6: *Simple demonstration of our Independent Edge Grouping Algorithm. Left: An example triangulated mesh with corresponding edge indices. Middle: After the first phase of the algorithm, all edges are listed with their nonadjacent edges arrays. Right: After the second phase of the algorithm, all the edges are grouped in six different groups such that none of the array members are connected with each other.*

In order to compare our modified Morse potential with the existing spring potentials and stretching constraint, we have implemented the Hookean [LBOK13] and STVK [RLK18] spring potentials as position based constraints along with the stretching constraint [MHHR07]. The mathematical definitions of those potentials are defined as:

$$C_{stretching}(x_1,x_2) = k(||x_2 - x_1|| - l_0) \tag{15}$$

$$C_{hookean}(x_1,x_2) = \frac{1}{2}k(||x_2 - x_1|| - l_0)^2 \tag{16}$$

$$C_{STVK}(x_1,x_2) = \frac{1}{2}k(||x_2 - x_1||^2 - l_0^2)^2 \tag{17}$$

where $k \geq 0$ is the spring stiffness, $x_1, x_2 \in \mathbb{R}^3$ are the spring endpoints and l_0 is the rest length.

We have tested the falling cloth example shown in Figure 1 under three different damping conditions. In the first case, we have not applied any damping to the global motion and observed that the cloth oscillates almost same with all constraints. One difference can be noted that Hookean spring behaves slightly softer than other springs. The relative errors can be seen in Figure 8. However, when we increase the damping coefficient slightly higher, stretching and STVK spring constraints damp the motion faster than Hookean spring and our modified Morse potential constraints. Furthermore, when we increase the damping notably, STVK spring and stretching constraints move almost identical and damp the motion in a peculiar way. On the other hand, Hookean spring and our modified Morse potential damp the motion more smoothly and still produce tiny and plausible wrinkles. The relative error plot supports this fact in Figure 7. In this example, the model consists of 4880 edges and 200 iterations have been applied for each simulation cycle due to gain high precision. The simulation performances have been observed as 6-7 FPS (frames per second) on CPU and 24-25 FPS on GPU.

In the flag example shown in Figure 2, the wind produce less wrinkles in stretching constraint than the other three spring constraints. This fact does not surprise us due to the fact that three spring potential constraints tend to produce more motion than the stretching constraint because of their potential formulations. Stretching constraint is approximated by the XPBD algorithm as if it is an actual potential energy function. In this flag example, the model consists of 4880 edges and 100 iterations have been applied. The simulation performances have been observed as 9-10 FPS on CPU and 34-35 FPS on GPU.

In another test case, we have tested a 3D volumetric triangular mesh model. We have coupled the volume preservation constraint with stretching and spring potential constraints. We hang the bunny model from the ears and observe that all three spring potentials behave more stretchy than the stretching constraint. Therefore, volume is better preserved with stretching constraint. On the other hand, our modified Morse potential constraint perform better volume preservation than the Hookean and STVK spring constraints that is shown in Figure 3. The change in damping does not effect this fact which can be seen in Figure 7. The bunny model consists of 12288 edges and 50 iterations have been applied. The simulation performances have been observed as 30-31 FPS on GPU.

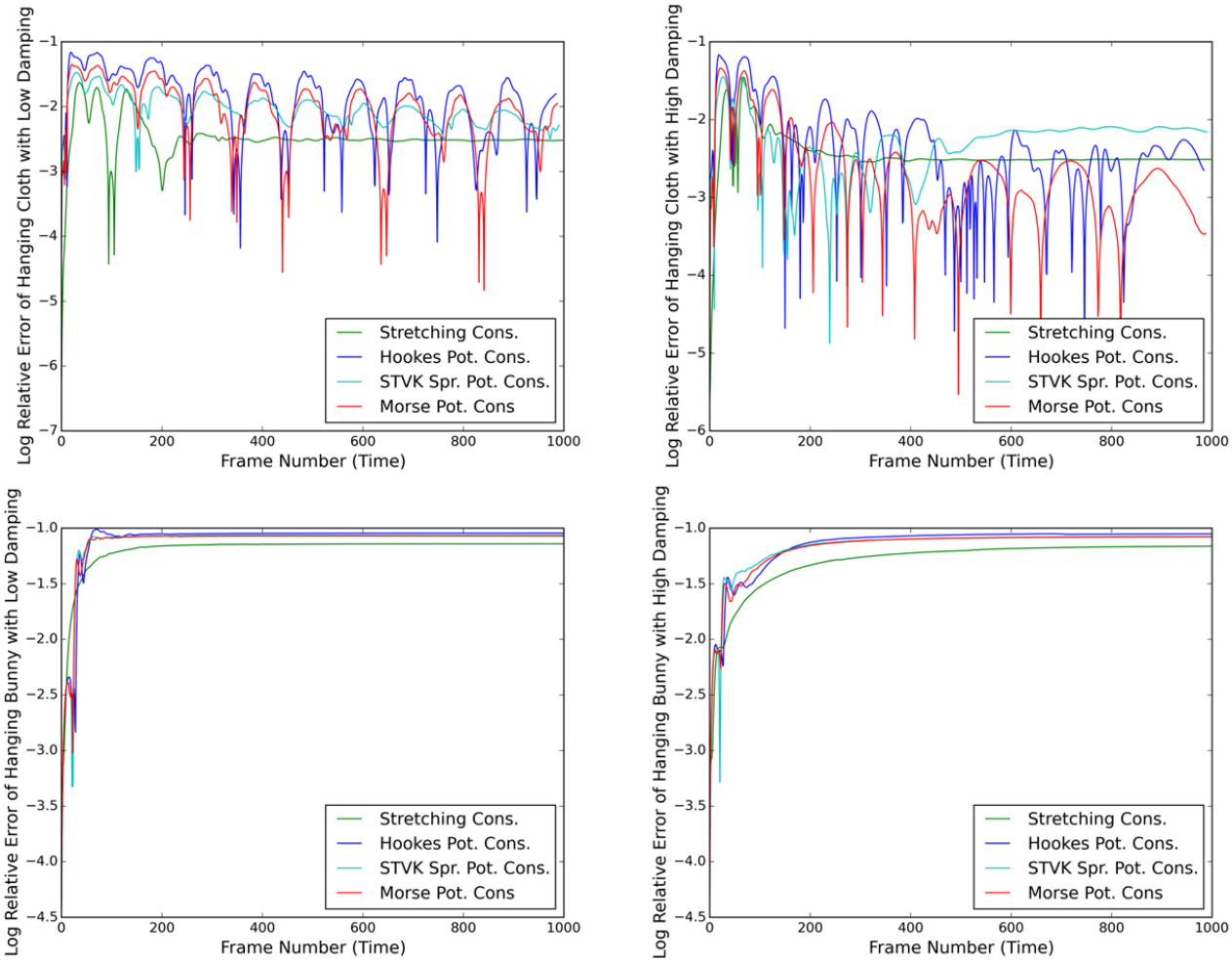

Figure 7: *The relative error of springs are compared. Top row shows the hanging cloth with damping coef.* $= 0.3$ *on left and damping coef.* $= 1.0$ *on right. The plots show that our modified Morse potential and Hookean spring can still produce wrinkles while the others damp the motion immediately. Bottom row shows the hanging bunny model with damping coef.* $= 0.3$ *on left and damping coef.* $= 1.0$ *on right. The plots show that our modified Morse potential and stretching constraint preserve the volume of the model better than the others. The relative error values are defined as,* $err = log(\frac{C}{I.V.})$*, where C represents the corresponding constraint, and I.V. represents the initial value of each corresponding primitive.*

We also tested the deformation behaviors under collision. When the cloth falls and contacts with the sphere as shown in Figure 5, all spring potentials almost behave similar to each other. Although there exist very slight differences, those can not be observed immediately. Besides, we have tested the collision on a 3D volumetric model and compared our proposed Morse potential - volume constraints coupling with STVK and Neo-Hookean finite element hyperelastic materials (FEM). We implemented those hyperelastic FEM materials from [BKCW14]. The results did not surprise us. Although FEM materials produce slightly more vivid reaction during the collisions, our modified Morse potential produces visually attractive results that can be seen in Figure 4. All collisions have performed in interactive rates on GPU.

8. Conclusion and Future Work

In this paper, we have modified the Morse potential function from molecular mechanics and presented it as an alternative to the existing spring potentials. We have adapted our modified Morse potential on parallel XPBD algorithm as a projected position constraint along with the existing spring potential functions such as Hookean and STVK springs. In order to maintain a stable parallelization, we developed an efficient and intuitive "Independent Edge Grouping" algorithm that provides a massive multi-thread computation on GPU. We compared the behavior of our proposed Morse potential with other springs as well as PBD's stretching constraint. We presented the strong and weak points of each spring constraint including our proposed technique. Although there is not any particular definition that indicates one spring model is better than the

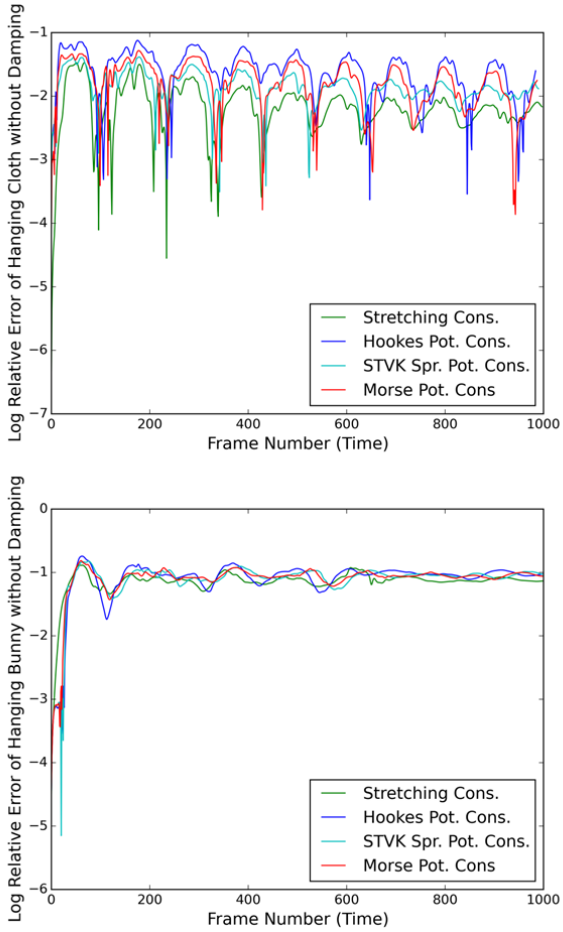

Figure 8: *The relative error of springs are compared without damping. Top row shows the hanging cloth. The plot shows that our modified Morse potential and Hookean spring move more dynamic than the others. Bottom row shows the hanging bunny. The plot shows that all spring potentials stretch to a certain end such that the volume preservation can be handled within the tolerable limits. The relative error values are defined as, $err = log(\frac{C}{I.V.})$, where C represents the corresponding constraint, and I.V. represents the initial value of each corresponding primitive.*

other, our modified Morse potential is a promising spring potential constraint that can be considered as an alternative to the existing spring potential functions. Besides, it is straightforward, stable and easy to adapt to the existing XPBD (or PBD) frameworks.

Currently, our implementation supports only simple static object collisions. In the future, we pursue to improve our collision handling with advanced methods similar to [MCKM15].

Acknowledgements

Author is grateful to Marco Fratarcangeli for insightful discussion, Keenan Crane for sharing "Duck Lifebuoy (Bob)" model. Bunny model is courtesy of Stanford University. This work is a result of the project NanoSTIMA (NORTE-01-0145-FEDER-000016), supported by Norte Portugal Regional Operational Programme (NORTE 2020), under the PORTUGAL 2020 Partnership Agreement, through the European Regional Development Fund (ERDF) and project UID/EEA/50008/2019.

References

[BHW94] BREEN D. E., HOUSE D. H., WOZNY M. J.: Predicting the drape of woven cloth using interacting particles. In *Proceedings of the 21st Annual Conference on Computer Graphics and Interactive Techniques* (1994), SIGGRAPH '94, pp. 365–372. 2

[BKCW14] BENDER J., KOSCHIER D., CHARRIER P., WEBER D.: Position-based simulation of continuous materials. *Computers & Graphics 44*, 0 (2014), 1 – 10. 1, 2, 8

[BMF03] BRIDSON R., MARINO S., FEDKIW R.: Simulation of clothing with folds and wrinkles. In *Proceedings of the 2003 ACM SIGGRAPH/Eurographics Symposium on Computer Animation* (2003). 2

[BML*14] BOUAZIZ S., MARTIN S., LIU T., KAVAN L., PAULY M.: Projective dynamics: Fusing constraint projections for fast simulation. *ACM Trans. Graph. 33*, 4 (jul 2014), 154:1–154:11. 5

[BMM17] BENDER J., MÜLLER M., MACKLIN M.: A survey on position based dynamics, 2017. In *EUROGRAPHICS 2017 Tutorials* (2017). 2, 4

[BW98] BARAFF D., WITKIN A.: Large steps in cloth simulation. In *Proceedings of the 25th Annual Conference on Computer Graphics and Interactive Techniques* (1998). 2

[CO16] CETINASLAN O., ORVALHO V.: Localized verlet integration framework for facial models. In *Articulated Motion and Deformable Objects - 9th International Conference, AMDO, Proceedings* (2016), vol. 9756 of *Lecture Notes in Computer Science*, Springer, pp. 1–15. 3

[DKWB18] DEUL C., KUGELSTADT T., WEILER M., BENDER J.: Direct position-based solver for stiff rods. *Computer Graphics Forum 37*, 6 (2018), 313–324. 3

[DS88] DAHL J. P., SPRINGBORG M.: The morse oscillator in position space, momentum space, and phase space. *The Journal of Chemical Physics 88*, 7 (1988), 4535–4547. 4

[Fau99] FAURE F.: Interactive solid animation using linearized displacement constraints. In *Computer Animation and Simulation '98* (Vienna, 1999), Springer Vienna, pp. 61–72. 2

[FP14] FRATARCANGELI M., PELLACINI F.: Towards a massively parallel solver for position based dynamics. In *Proceedings of SIGRAD, Visual Computing, Göteborg, Sweden* (2014), no. 106, pp. 25–31. 3

[FP15] FRATARCANGELI M., PELLACINI F.: Scalable partitioning for parallel position based dynamics. *Computer Graphics Forum* (2015). 6

[FTP16] FRATARCANGELI M., TIBALDO V., PELLACINI F.: Vivace: A practical gauss-seidel method for stable soft body dynamics. *ACM Trans. Graph. 35*, 6 (Nov. 2016), 214:1–214:9. 3, 6

[GG94] GASCUEL J.-D., GASCUEL M.-P.: Displacement constraints for interactive modeling and animation of articulated structures. *The Visual Computer 10*, 4 (Apr 1994), 191–204. 2

[GHDS03] GRINSPUN E., HIRANI A. N., DESBRUN M., SCHRÖDER P.: Discrete shells. In *Proceedings of the 2003 ACM SIGGRAPH/Eurographics Symposium on Computer Animation* (2003). 2

[GHF*07] GOLDENTHAL R., HARMON D., FATTAL R., BERCOVIER M., GRINSPUN E.: Efficient simulation of inextensible cloth. *ACM Trans. Graph. 26*, 3 (2007), 49. 2

[GW59] GIRIFALCO L. A., WEIZER V. G.: Application of the morse potential function to cubic metals. *Phys. Rev. 114* (May 1959), 687–690. 4

[Jak01] JAKOBSEN T.: Advanced character physics. In *In Proceedings of the Game Developers Conference* (2001), pp. 383–401. 2

[KCMF12] KIM T.-Y., CHENTANEZ N., MÜLLER-FISCHER M.: Long range attachments - a method to simulate inextensible clothing in computer games. In *Proceedings of the ACM SIGGRAPH/Eurographics Symposium on Computer Animation* (2012), pp. 305–310. 2

[LBOK13] LIU T., BARGTEIL A. W., O'BRIEN J. F., KAVAN L.: Fast simulation of mass-spring systems. *ACM Trans. Graph. 32*, 6 (nov 2013), 214:1–214:7. 2, 7

[MÖ8] MÜLLER M.: Hierarchical position based dynamics. In *Workshop in Virtual Reality Interactions and Physical Simulation "VRIPHYS" (2008)* (2008). 3

[Mar12] MARCO F.: Position based facial animation synthesis. *Computer Animation and Virtual Worlds 23*, 3-4 (2012), 457–466. 2

[MC10] MÜLLER M., CHENTANEZ N.: Wrinkle meshes. In *Proceedings of the 2010 ACM SIGGRAPH/Eurographics Symposium on Computer Animation* (2010), pp. 85–92. 2

[MC11] MÜLLER M., CHENTANEZ N.: Solid simulation with oriented particles. *ACM Trans. Graph. 30*, 4 (2011), 92:1–92:10. 2

[MCKM14] MÜLLER M., CHENTANEZ N., KIM T.-Y., MACKLIN M.: Strain based dynamics. In *Eurographics/ ACM SIGGRAPH Symposium on Computer Animation* (2014). 2, 3

[MCKM15] MÜLLER M., CHENTANEZ N., KIM T.-Y., MACKLIN M.: Air meshes for robust collision handling. *ACM Trans. Graph. 34*, 4 (jul 2015), 133:1–133:9. 9

[MHHR07] MÜLLER M., HEIDELBERGER B., HENNIX M., RATCLIFF J.: Position based dynamics. *Journal of Visual Communication and Image Representation 18*, 2 (2007), 109–118. 1, 2, 3, 5, 7

[MKC12] MÜLLER M., KIM T.-Y., CHENTANEZ N.: Fast Simulation of Inextensible Hair and Fur. In *Workshop on Virtual Reality Interaction and Physical Simulation* (2012), The Eurographics Association. 2

[MM13] MACKLIN M., MÜLLER M.: Position based fluids. *ACM Trans. Graph. 32*, 4 (jul 2013), 104:1–104:12. 2

[MMC16] MACKLIN M., MÜLLER M., CHENTANEZ N.: Xpbd: Position-based simulation of compliant constrained dynamics. In *Proceedings of the 9th International Conference on Motion in Games* (2016), MIG '16, ACM, pp. 49–54. 1, 2, 3, 5, 6

[Mor29] MORSE P. M.: Diatomic molecules according to the wave mechanics. ii. vibrational levels. *Phys. Rev. 34* (Jul 1929), 57–64. 1, 3, 4

[NMK*06] NEALEN A., MÜLLER M., KEISER R., BOXERMAN E., CARLSON M.: Physically based deformable models in computer graphics. *Comput. Graph. Forum 25*, 4 (2006), 809–836. 2

[RLK18] ROJAS J., LIU T., KAVAN L.: Average vector field integration for st. venant-kirchhoff deformable models. *IEEE Transactions on Visualization and Computer Graphics* (2018). 1, 7

[SLF08] SELLE A., LENTINE M., FEDKIW R.: A mass spring model for hair simulation. *ACM Trans. Graph. 27*, 3 (2008), 64:1–64:11. 2

[SLM06] SERVIN M., LACOURSIERE C., MELIN N.: Interactive simulation of elastic deformable materials. In *In proceedings of Sigrad Conference* (2006), pp. 22–32. 2

[Sta09] STAM J.: Nucleus: Towards a unified dynamics solver for computer graphics. In *Computer-Aided Design and Computer Graphics, 2009. CAD/Graphics '09. 11th IEEE International Conference on* (2009), pp. 1–11. 2

[TNGF15] TOURNIER M., NESME M., GILLES B., FAURE F.: Stable constrained dynamics. *ACM Trans. Graph. 34*, 4 (jul 2015), 132:1–132:10. 2

[TPBF87] TERZOPOULOS D., PLATT J., BARR A., FLEISCHER K.: Elastically deformable models. *SIGGRAPH Comput. Graph. 21*, 4 (1987), 205–214. 2

[USS14] UMETANI N., SCHMIDT R., STAM J.: Position-based elastic rods. In *Proceedings of the ACM SIGGRAPH/Eurographics Symposium on Computer Animation* (2014), SCA '14, pp. 21–30. 3

[Wan15] WANG H.: A chebyshev semi-iterative approach for accelerating projective and position-based dynamics. *ACM Trans. Graph. 34*, 6 (2015), 246:1–246:9. 3