CEIG - Spanish Computer Graphics Conference (2018)
I. García-Fernández and C. Ureña (Editors)

Muscle Simulation with Extended Position Based Dynamics

M. Romeo, C. Monteagudo and D. Sánchez-Quirós

El Ranchito Imagen Digital

Abstract

Recent research on muscle simulation for Visual Effects relies on numerical methods such as the Finite Element Method or Finite Volume Method. These approaches produce realistic results, but require high computational time and are complex to set up. On the other hand Position Based Dynamics offers a fast and controllable solution to simulate surfaces and volumes, but there is no literature on how to implement constraints that could be used to realistically simulate muscles for digital creatures with this method.

In this paper we extend the current state-of-the-art in Position Based Dynamics to efficiently compute realistic skeletal-muscle simulation. In particular we embed muscle fibers in the solver by adding an anisotropic component to the distance constraints between mesh points and apply overpressure to realistically model muscle volume changes under contraction.

We also present a technique that consistently provides an internal structure for our muscle volumes. We use this structure to preserve the shape and extract relevant information for the activation of the muscle fibers.

Finally, we propose a modification of the Extended Position Based Dynamics algorithm and describe other details for proper simulation of character's muscle dynamics.

CCS Concepts

•***Computing methodologies*** → ***Computer graphics;*** •***Applied computing*** → ***Physics; Media arts;***

1. INTRODUCTION

Digital characters have become prominent in visual effects and computer animation. With the increasing overall quality of computer generated imagery, the required level of realism of the character's dynamics has also increased drastically. Standard linear or dual-quaternion skinning solutions are not enough to represent the level of realism required by today visual effects. For this reason, the simulation of character's muscles is becoming vastly used to produce fine details, preserve volumes, make deformations and animations more believable and convey strength through movement.

The main requirement for a muscle simulation system is to produce high quality results with proper interaction between muscles, believable activation of muscle fibers and rendition of muscle dynamics. However a production-ready system also has to fulfill the following requirements:

- **Fast convergence of simulation** is necessary to review results as soon as possible, implement eventual feedback and execute as many iterations as possible.
- **Easy to setup** systems allow companies to ease the entire workflow, as the processes prior to running the simulation do not get overcomplicated because of the requirements of the simulation system.
- **Intuitive controls** allow artists to focus on what is really important, without having to fiddle with complex parameters. These parameters should produce predictable results.
- **Artistic control** enables supervisors and directors to make changes that comply with their vision, even if not completely realistic or physically correct.

Most simulation systems for digital characters as Ziva VFX [zivc], Weta's Tissue [tis] or ILM's Zeno [CBC15] rely on Finite Element Method (FEM) and Finite Volume Method (FVM). Unfortunately FEM and FVM, while providing excellent results, are known for not complying with the requirements listed in the previous paragraph. Instead, Extended Position Based Dynamics (XPBD) [MMC16] offers a better compromise between quality and the other, more practical, requirements.

In this paper we present an XPBD-based system capable of efficiently simulating realistic digital characters' muscles. The core contribution is the formulation of a fibrous constraint as a modification of the standard Position Based Dynamics (PBD) distance constraint, allowing muscles to contract during simulation. We describe how the volumetric structure of muscles and their fibers' flow is automatically computed and provide a description of how we set up a complete human muscle rig for testing. Finally we present how we modified the original PBD and XPBD algorithm to allow the simulation of the muscle rig.

DOI: 10.2312/ceig.20181146

2. RELATED WORK

Research in modeling and simulating muscle systems has relied on different approaches through the years. In their survey on muscle modeling and simulation for computer graphics [LGK*12], Lee et al. provide a broad overview of proposed mechanical models, geometric, physically-based and data-driven approaches. Muscles are not simple in shape and structure, in fact muscles could present multiple tendons, several points of attachment and fibers not flowing parallel to each other. It is noted that this complexity cannot be properly represented through geometrically-based approaches and require more sophisticated solutions. Different approaches to simulate muscle dynamics are discussed, but there is no mention of any solution using Position Based Dynamics.

Since early mechanical models, there was a clear understanding that the internal forces produced by muscle fibers were a key factor to the behavior of a contracting muscle [GH24]. More recent mechanical models also stress the importance of the direction of such fibers [ZM89].

Most of the current muscle simulation systems in the visual effects industry are based on FEM or FVM [TBHF03, TSIF05, tis, JBE*16, hou]. Mixed approaches blending procedural deformation and physical simulation have also been used [CBC15, MMT*16].

To overcome the difficult setup process required by FEM and FVM, companies and researchers have implemented specific tools [ziva, SSW*10], allowing artists to implement one muscle rig template and transfer it across similar creatures. Another solution is to provide artists with modeling tools designed to produce simulation-ready geometries [TRF16]. Still, the process remains cumbersome, with some companies relying on MRI scans of real animals or people [Lan17]. Moreover, FEM and FVM do not allow for volume alteration during simulation, while real muscles' volume increases during activation [NQMT91].

Recent work from Müller et al. [MHHR07] and Macklin et al. [MMCK14, MMC16] on PBD and XPBD shows that it is possible to build a comprehensive framework for physical simulation capable of simulating different interacting elements (e.g. soft bodies, rigid bodies, fluids and gases). By operating over point positions and relying on a Gauss-Seidel-type iteration, PBD advantages include controllability and simplicity, together with high quality results and speed. XPBD solves one major issue related with PBD by making the stiffness not dependent from the number of iterations or time-steps making it completely suitable for production.

PBD is making its entry into the animation industry [SLG14], but there is still little literature about muscle simulation with PBD [Fra12] and no reference to any solution that integrates fibers' activation into the simulation loop.

3. FIBER-EMBEDDED CONSTRAINTS

As seen in section 2, there is a lack of comprehensive literature on using PBD for muscle simulation. One of the first issues to overcome was that PBD is commonly intended to work on geometry's surface, which makes shape preservation difficult during compression produced by collisions. To overcome this issue we defined a methodology to build an internal muscle structure to better preserve the shape and volume of the muscle. Furthermore, we studied how to compute fiber directions for the muscles and modify the distance constraint defined in [MHHR07] to embed the fibers' activation and robustly simulate muscle contraction.

3.1. The Internal Muscle Structure

In order to build a coherent internal structure the user manually selects the *root vertices*: a set of vertices, typically located on the surface around the barycenter of the mesh, from which fibers should propagate flowing towards the tendons. Then the geometry of each muscle is processed to define its flow and branching. This is achieved in three phases: *traverse*, *re-sample* and *connect*.

Traverse. Starting from the manually-selected set of root vertices, the mesh vertices are iteratively traversed by following the mesh connectivity. The process produces a set of samples stored in a tree structure resembling the traversed mesh branching (Figure 1). For each sample the set of traversed vertices is stored.

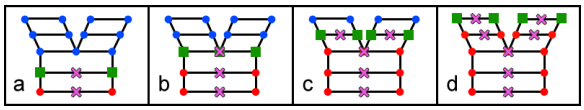

Figure 1: *Four traversing iterations of a mesh. a) For each visited set of vertices (red dots) a sample is generated as the average position of the vertices (magenta cross). b) The connected vertices are selected (green square) from the non traversed points (blue dots) and a new sample is generated. c) The connected vertices are made by two separated sets of vertices that do not share any edge, determining a branching point and two samples. d) For the two detected branches the traversing continues independently.*

Re-sample. For each branch, the samples are reduced to fit a predefined maximum distance between points. For each re-sampled point, the set of relative vertices is also rearranged.

Connect. Constraints are created between each sample and its children. Constraints are also generated between each sample and corresponding surface vertices.

The resulting internal structure for a biceps muscle modeled from scan data (see Figure 2), resembles the fibers direction inside the muscle. Thanks to this, it will behave optimally when contracted and provide the information required to define the fibers flow on the surface of the muscle.

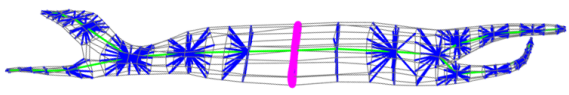

Figure 2: *The biceps structure: fibers' root vertices (purple), the internal branches (green) and the connections between internal branches and surface vertices (blue). The resulting internal branches resemble the direction of the internal fibers of the muscle.*

This approach works independently from the mesh topology and

density. In Figure 3 we show the result obtained by generating the internal structure of the original scan data for the biceps muscle (26176 triangles).

One important aspect of our approach is that the obtained internal branching resembles the internal flow of muscle fibers. This differs from the tetrahedralization process required by finite element approaches, which tend to produce complex volume meshes that do not resemble the structure and fibers' flow of muscles (see Figure 4).

In the next sub-section we will describe how we use this information to compute the direction of the muscle fibers.

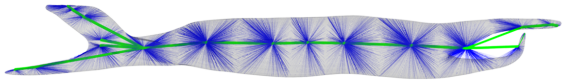

Figure 3: *The structure generated on the original biceps scan data: the internal branches (green) and the connections between internal branches and surface vertices (blue).*

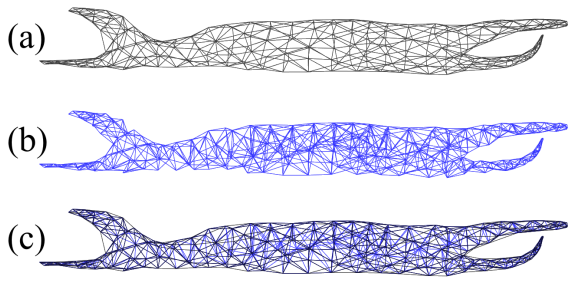

Figure 4: *The structure generated by a Delaunay tetrahedralization algorithm for use with Houdini's FEM muscle system [hou]: the surface mesh triangles (a), the internal tetrahedra edges (b) and the complete volume mesh (c). The structure is apparently random and does not resemble the flow of fibers at all, with a high number of edges being generated.*

3.2. Computing the Fibers' Flow

Having computed the internal structure of the muscle, we use this information to extrapolate the muscle fiber direction at each point of its geometry. The idea is to use the internal structure as a guide to direct the muscle fibers; this makes sense as the internal branches are built in a way that describes the flow of the superficial mesh and mimic the typical distribution of fibers.

Given a point $\mathbf{b}_i$ on the internal muscle branch, we first compute the fiber vector δ_i at this point as the normalized distance vector from $\mathbf{b}_i$ to its parent point $\mathbf{b}_{i+1}$ in the hierarchy of the internal branch. Then, for each vertex $\mathbf{p}_j$ in the set of connected surface vertices of $\mathbf{b}_i$, we compute the fiber vector $\mathbf{f}_j$ as the average of δ_{i-1}, δ_i and δ_{i+1} weighted by the distance between $\mathbf{p}_j$ and $\mathbf{b}_i$, $\mathbf{b}_{i-1}$ and $\mathbf{b}_{i+1}$. Finally, we project $\mathbf{f}_j$ onto the surface surrounding the vertex

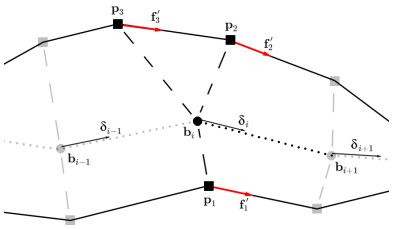

Figure 5: *For a point $\mathbf{b}_i$ on the muscle's internal branch (dotted line), with adjacent branch points being $\mathbf{b}_{i-1}$ and $\mathbf{b}_{i+1}$, we compute the fibers' flow of the points $\mathbf{p}_1$, $\mathbf{p}_2$ and $\mathbf{p}_3$ ($\mathbf{f}'_1$, $\mathbf{f}'_2$ and $\mathbf{f}'_3$ respectively) which are connected to $\mathbf{b}_i$ by the internal structure (dashed line).*

itself obtaining the final fiber vector $\mathbf{f}'_j$ for the vertex $\mathbf{p}_j$ (see Figure 5).

The result is a consistent flow that naturally travels from the mid-section (root vertices) of the muscle towards its tendons, including possible branching as in the biceps muscle (see Figure 6).

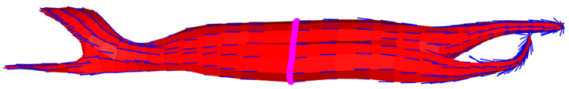

Figure 6: *Fibers flow on a biceps muscle: fibers' root vertices (purple) and the fibers' flow (blue lines).*

3.3. Activation-Driven Fibrous Constraints

In [MHHR07], the authors define the formulas for the projection of distance constraints between two points $\mathbf{p}_1$ and $\mathbf{p}_2$ as in Eqs. 1 and 2. Here $w1$ and $w2$ are the correction weights for $\mathbf{p}_1$ and $\mathbf{p}_2$ respectively, defined as the inverse of their masses. The distance constraint C is defined in Eq. 3 as the adjustment required to achieve a goal distance d between $\mathbf{p}_1$ and $\mathbf{p}_2$.

$$\Delta\mathbf{p}_1 = -\frac{w1}{w1+w2} C \frac{\mathbf{p}_1 - \mathbf{p}_2}{|\,\mathbf{p}_1 - \mathbf{p}_2\,|} \tag{1}$$

$$\Delta\mathbf{p}_2 = +\frac{w2}{w1+w2} C \frac{\mathbf{p}_1 - \mathbf{p}_2}{|\,\mathbf{p}_1 - \mathbf{p}_2\,|} \tag{2}$$

$$C = |\,\mathbf{p}_1 - \mathbf{p}_2\,| - d \tag{3}$$

During activation, the length of a muscle reduces along the fibers' direction. This idea contrasts the idea of distance constraints that are, in fact, trying to preserve the distance between mesh points. We propose to extend the concept of the distance constraint by being able to alter the goal distance between the two points according to the activation of muscle fibers. This way, we could reduce the desired distance between points along the fiber's direction when the muscle is activated.

For each pair of constrained points $\mathbf{p}_1$ and $\mathbf{p}_2$ with fiber directions $\mathbf{f}'_1$ and $\mathbf{f}'_2$ respectively, we get the normalized vector of the sum of the two fiber vectors and obtain ϕ by computing the absolute value of its components (see Eq. 4).

$$\phi = \text{abs}\left(\frac{\mathbf{f}'_1 + \mathbf{f}'_2}{|\mathbf{f}'_1 + \mathbf{f}'_2|}\right) \tag{4}$$

To obtain the activated edge distances, we compute the distance vector between the two points $\mathbf{p}_1$ and $\mathbf{p}_2$ at rest ($\mathbf{r}_1$ and $\mathbf{r}_2$ respectively). We obtain the amount of fiber compression in each coordinate for the edge connecting $\mathbf{r}_1$ and $\mathbf{r}_2$ by subtracting the vector ϕ multiplied by the scalar activation value a from a vector of ones ($\overrightarrow{\mathbf{1}}$). The activation value a relates to the level of fibers' activation in the muscle. Performing a component-wise multiplication between the distance vector and the fiber compression vector gives us the new edge vector after the activation of muscle fibers. The norm of this new vector is our new goal distance, which we call σ (Eq. 5).

$$\sigma = \mid \left((\mathbf{r}_1 - \mathbf{r}_2) \circ \left(\overrightarrow{\mathbf{1}} - \phi a\right)\right) \mid \tag{5}$$

The value σ will work if we simulate muscles that are completely relaxed ($a = 0$) at rest. However any living being's muscles are activated and contracted, even in rest pose. Because of this, it is necessary to compute a new value σ' that lets us redefine the goal distance by taking into consideration a rest activation value a_0.

In Eq. 6 we define ε by inverting the operation from Eq. 5, but using a_0 instead of a. This provides a new rest distance vector without any activation.

$$\varepsilon = \frac{\mathbf{r}_1 - \mathbf{r}_2}{\overrightarrow{\mathbf{1}} - \phi a_0} \tag{6}$$

Now that we have the real rest distance vector ε we can rewrite Eq. 5 as Eq. 7, which will give us the distance σ' between the two points at rest without activation.

$$\sigma' = \mid \left(\varepsilon \circ \left(\overrightarrow{\mathbf{1}} - \phi a\right)\right) \mid \tag{7}$$

Finally, we substitute d from Eq. 3 with σ', obtaining C' (Eq. 8) and then substitute it in Eqs. 1 and 2, getting Eqs. 9 and 10 respectively.

$$C' = \mid \mathbf{p}_1 - \mathbf{p}_2 \mid - \sigma' \tag{8}$$

$$\Delta\mathbf{p}_1 = -\frac{w1}{w1 + w2} C' \frac{\mathbf{p}_1 - \mathbf{p}_2}{\mid \mathbf{p}_1 - \mathbf{p}_2 \mid} \tag{9}$$

$$\Delta\mathbf{p}_2 = +\frac{w2}{w1 + w2} C' \frac{\mathbf{p}_1 - \mathbf{p}_2}{\mid \mathbf{p}_1 - \mathbf{p}_2 \mid} \tag{10}$$

In order to avoid instabilities and divisions by zero, a and a_0 have to be smaller than one.

As the differences between PBD and XPBD do not affect how constraints work, we can easily use this in both PBD and XBPD [MMC16]. In Figure 7 we show three different states for a biceps muscle using our fibrous constraint, displaying the difference between the muscle at rest, activated and with volume-gain during activation.

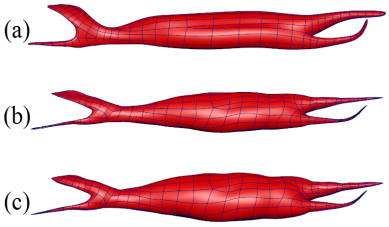

Figure 7: *Three different states of a biceps with fibrous constraints: the biceps at rest (a), the biceps with fibers activated and volume preservation(b) and the biceps fibers activated with volume gain applied (c).*

4. MUSCLE DYNAMICS

4.1. Solver Setup

Our solver uses volume constraints to preserve and alter (overpressure) volume, fibrous constraints to contract the muscles when activated and a set of external constraints to connect muscles and skeleton meshes during simulation. We decided not to use bending constraints into our solver as we wanted the muscles to have complete freedom of deformation when not active. The overall shape of the muscle is preserved by its internal structure and when muscle fibers activate the increased stiffness and contraction produce a more rigid dynamic. By not using bending constraints, we also reduce the total count of constraints to be computed.

During the activation of the muscle fibers, we alter the stiffness property of the material from relaxed (5×10^5) to active (5×10^7) in order to mimic the behavior of real muscle tissue. Furthermore, as muscles gain volume during contraction [NQMT91], we used overpressure [MHHR07] and made a correlation between the activation of muscle fibers and the goal volume of the muscle itself (see Figure 17) so that muscles' volume increases as they get more active. The artist, together with supervisors can define how the volume gain is affected by the activation through a curve that maps these two values. This results in a realistic behavior of muscles becoming stiffer and bigger during contraction, but these parameters can also be tweaked to obtain non-realistic results for fantastic creatures.

4.2. Full-Body Simulation

To demonstrate the fit for purpose of our system, we created a full human skeleton-muscle system (see Figure 8): one skeleton envelope (10728 triangles) and 140 muscles (37930 triangles). The

skeleton envelope is a one-piece mesh modeled over a high resolution geometry of skeleton bones. This provides a continuous surface that helps us avoiding artifacts when muscles slide over the skeleton geometry.

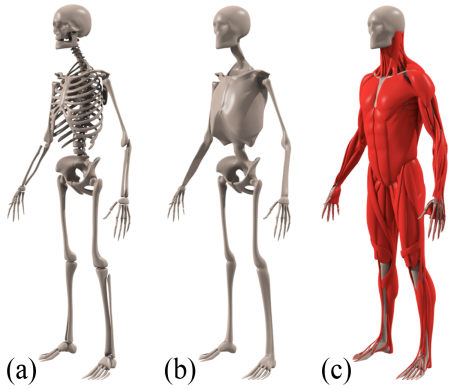

Figure 8: *The full human skeleton-muscle system: the high resolution skeleton bones (a); the skeleton envelope modeled on top of the high resolution bones to provide a continuous and lower resolution mesh for constraints (b); the 140 separate muscles over the skeleton envelope (c).*

We simulated the human muscles using motion capture data of an actor walking and running (see Figure 9) and other test motions like forearm torsion, elbow bending (see Figure 10) and arm shaking. The motion data was transferred from the animation joints to the skeleton bones using standard linear skinning techniques, assigning to each bone only one influence joint of the animation skeleton (a rigid deformation of each bone by only one joint). The deformed skeleton bones are then used to wrap-deform the skeleton envelope, which is then used as attachment and collider for the muscles simulation.

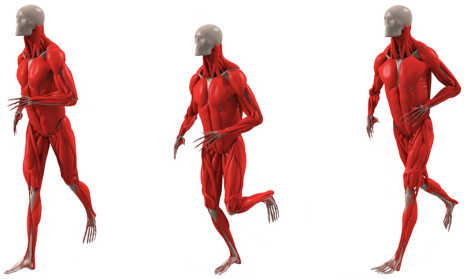

Figure 9: *Three frames from the simulation of running motion data.*

To drive the activation of the muscles we developed linear and angular velocity sensors that were attached to the animation joints to detect velocity and acceleration of translating and rotating joints. We also developed a trigger system that contracts the muscle whenever the activation level surpasses a threshold and relaxes it through time when the threshold is not met; this was used to detect rapid decelerations due to feet making contact with the ground during the run motion, activate the muscles and slowly relaxing them.

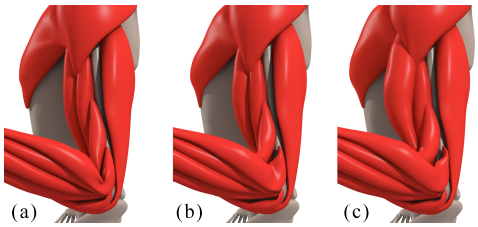

Figure 10: *Arm bending: Standard XPBD with standard distance constraints (a), fibers contraction (b) and with the addition of volume gain (c). It is important to notice the behavior of the inner part of the elbow: in (a) the muscles are simply bending and folding onto each other; using our fibrous constraint in (b) and (c) it is possible to activate and contract the muscles, creating a more realistic shape.*

The set of muscles included planar, fusiform and pennate muscle types [LGK*12]. The process to automatically generate the internal muscle structure and define the fibers' direction, as well as the contraction of the fibers worked properly for all types of muscles.

4.3. External Constraints and Attach Points

One key element of a muscle system is the fascia, an elastic connective tissue that lays between muscles (deep) and contains all the muscles together (superficial). The fascia is important to determine how muscles slide between each other and to compress muscles (and organs) together around the skeleton. We emulated the effect of deep and superficial fascia by creating what we call *external constraints* (see Figure 11), a mixture of muscle-to-muscle and muscle-to-skeleton sliding, soft and hard constraints. This made muscles interact between each other and act as a single system. This setup produced 135 780 dynamic constraints, including mesh (75 644), internal structure (20 572), external (39 424) and volume (140).

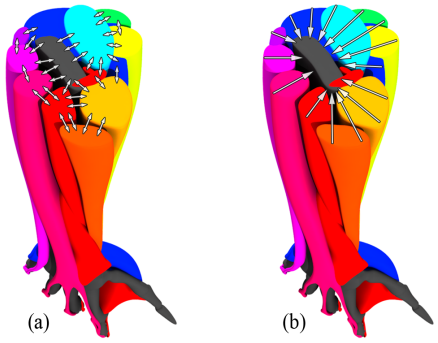

Figure 11: *A view of a section of the right arm. A series of soft, hard and slide constraints are created between muscles (colored) and skeleton (dark gray). Muscle-to-muscle and muscle-to-skeleton constraints are created between surfaces with opposing normals to emulate the effect of deep fascia (a). Muscle-to-skeleton constraints are created between muscle surfaces at the outer boundary of the arm and the skeleton to emulate the containing effect of the superficial fascia (b).*

The system can connect each muscle point to one or multiple target meshes to allow muscle vertices to interact with multiple other muscles. In addition, muscles have a minimum desired distance to their targets that overrides the correction distance in the case such minimum distance is not fulfilled.

The adjustment for each vertex produced by the external constraints is computed interpolating the resulting positions from the following three constraint types:

- **Sliding**: the constrained point maintains the initial distance to the target mesh, sliding along the target surface. If the muscle has more than one target, we allow the constraint to change its target dynamically during the simulation. The sliding surface is determined by searching for the closest surface point to the constrained point along all of its target meshes.
- **Soft**: the constrained point maintains the initial distance to the closest point on the target mesh. In contrast to sliding constraints, soft cannot dynamically switch from one target mesh to another. On initialization the closest point is computed between the constrained point and all its target meshes. The resulting closest point is stored by means of barycentric coordinates; this is later used to obtain its world position at every frame in the simulation.
- **Hard**: the constrained point maintains its relative position to the surface of its target mesh. On initialization we compute the transformation of the constrained point relative to its closest point on the target mesh. This is achieved by storing the position of the constrained vertex relative to the tangent space of its closest point.

We interpolate the external constraints taking into account per-vertex weight values defined by the artist for each type of constraint (slide, soft and hard). These weights are used to interpolate the position correction computed by each constraint and obtain one single position to project the vertex to. Given the input raw values w_{slide}, w_{soft} and w_{hard} for sliding, soft and hard weights respectively such that $w_{slide}, w_{soft}, w_{hard} \in \{0,1\}$, we compute the final weights w'_{slide}, w'_{soft} and w'_{hard} as in Eqs. 11, 12 and 13. Sliding is especially important to model the behavior of deep and superficial fascia, thus we do not alter the weight of the sliding constraint, making it more relevant to the final interpolation.

$$w'_{slide} = w_{slide} \tag{11}$$

$$w'_{soft} = (1 - w'_{slide}) \left(\frac{w_{soft}}{w_{soft} + w_{hard}} \right) \tag{12}$$

$$w'_{hard} = (1 - w'_{slide}) \left(\frac{w_{hard}}{w_{soft} + w_{hard}} \right) \tag{13}$$

For each vertex, once all projections are computed for each constraint with weight higher than zero, we interpolate these results to obtain our final constraint projection $\Delta \mathbf{p}_{ext}$ for the external constraints (Eq. 14).

$$\Delta \mathbf{p}_{ext} = \Delta \mathbf{p}_{slide} w'_{slide} + \Delta \mathbf{p}_{soft} w'_{soft} + \Delta \mathbf{p}_{hard} w'_{hard} \tag{14}$$

Finally, we also attach vertices straight to the animation joints. This is easily done, thanks to the solution provided in [MHHR07] for attachments to kinematic objects. These attachments are particularly useful to pinpoint vertices at the extremities of tendons to the animated joints. The vertices with attachment points are ignored by the solver and forced to the indicated target position. This allows the muscles to follow the animation as much as possible, improving the quality of the result.

5. ALGORITHM

In Algorithm 1 we show how we modified the PBD [MHHR07] and XPBD [MMC16] algorithms to account for the muscle activation. Lines 1 – 4 build the system by initializing the positions, velocities and weights for each vertex, the constraints, and also the muscle fibers. Lines 5 – 18 implement the simulation loop to compute semi-implicit Euler integration of the velocities (6 – 9), project the constraints (10 – 14) and update velocities and positions (15 – 18). As in the original PBD algorithm, we use $\mathbf{x}_i$ for the positions at the beginning of the loop, $\mathbf{p}_i$ for the predicted positions used in the solver and $\mathbf{v}_i$ for the velocities.

Algorithm 1 Muscle initialization and simulation loop

```
1:  for all muscles m do
2:      for all vertices i do initialize x_i, v_i, w_i
3:      initialize fibers
4:      initialize constraints
5:  loop
6:      for all muscles m do
7:          for all vertices i do v_i ← v_i + Δt w_i f_ext(x_i)
8:          damp velocities
9:          for all vertices i do p_i ← x_i + Δt v_i
10:     for i in globalMaxIterations do
11:         for all muscles m do
12:             if i < m.maxIterations then
13:                 update externalConstraints
14:                 for all constraints c do c.projectConstraint
15:     for all muscles m do
16:         for all vertices i do
17:             v_i ← (p_i − x_i)/Δt
18:             x_i ← p_i
```

As different muscles might require a different number of iterations to converge satisfactorily, we also included the possibility to override the iterations count per-muscle, which is possible thanks to the structure of the simulation loop based on a global count of the entire system (line 10) and a maximum count per muscle (line 12).

To properly integrate the computation of all external constraints together with all fibrous constraints, the algorithm progresses by computing all constraints for each muscle at each iteration first, then moves to the next iteration.

6. CONCLUSIONS AND FUTURE WORK

We extended XPBD to efficiently simulate complex muscle systems. We also found that the modeling of the muscle rig is very

simple as PBD-ready geometry can be prepared in little time as, even if the geometry presented intersections at rest, the proposed solution worked properly and did not produce artifacts. The system is also highly controllable, thanks to the advantages of using PBD.

The generation of the internal structure and fibers' flow can be achieved in one single step and with very little human input by simply selecting a subset of the mesh vertices near the barycenter of the muscle geometry. The proposed approach worked satisfactorily for all the 140 muscles created for our full-body human muscle rig, which presented a broad range of planar, fusiform and pennate muscles (see Figures 12, 13, 14).

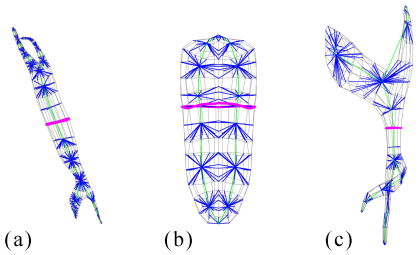

Figure 12: *Internal structure of three muscles: root vertices (purple), internal branches (green) and connections (blue).*

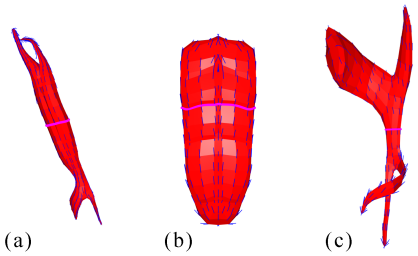

Figure 13: *Fibers' flow of three muscles: fibers' root vertices (purple) and the fibers' flow (blue lines).*

We developed a plug-in for Autodesk Maya [may] to evaluate and use our work in production. The plug-in enables artists to modify the behavior of sensors and muscles through a series of attributes and curves that influence the level of activation of muscles (see Figure 15), which provide the required level of creative control. The intricate configuration of solvers, muscles, constraints and sensors can be comfortably managed using the in-built node editor within Autodesk Maya (see Figure 16).

We distributed the simulation of a full human body (135 780 constraints) into six processes, achieving a performance of 44.04 seconds per frame (10 iterations per frame) on an Intel Core i7-4790K (4.00 GHz) with 32 GB of RAM. This compares well with other optimized FEM-based solutions as Ziva VFX: 37.6 seconds per frame (30 036 Tetrahedra), obtained on equal hardware resources [zivb].

All the performed tests and the ongoing use of the system in production are showing that the system consistently produces believable results with the required ease of setup and use. It also shows

Figure 14: *Internal branching computed on the full set of muscles. The green lines represent the internal branches of the muscles.*

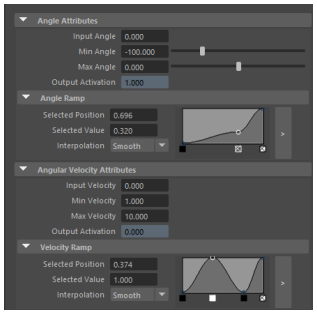

Figure 15: *A snapshot of the Maya attribute editor window. This shows the level of controllability of our rotation sensor that outputs the activation level produced by the angle, angular velocity and acceleration. Most of the parameters can be configured to work in a non-linear manner by editing parameters' curves.*

that the internal structure and fibers' flow computation is consistently adequate.

Future work will focus on the parallelization of the simulation system to further improve performance. We are also working on using the system on the superficial fascia and fat layers, make a clear distinction between muscles and tendons, while further investigating the anisotropic and non-linear properties of muscles.

Acknowledgement

We would like to thank Tamara Meirás, Pablo Hernández M. and Javier Mansilla at El Ranchito for their valuable support and constant feedback.

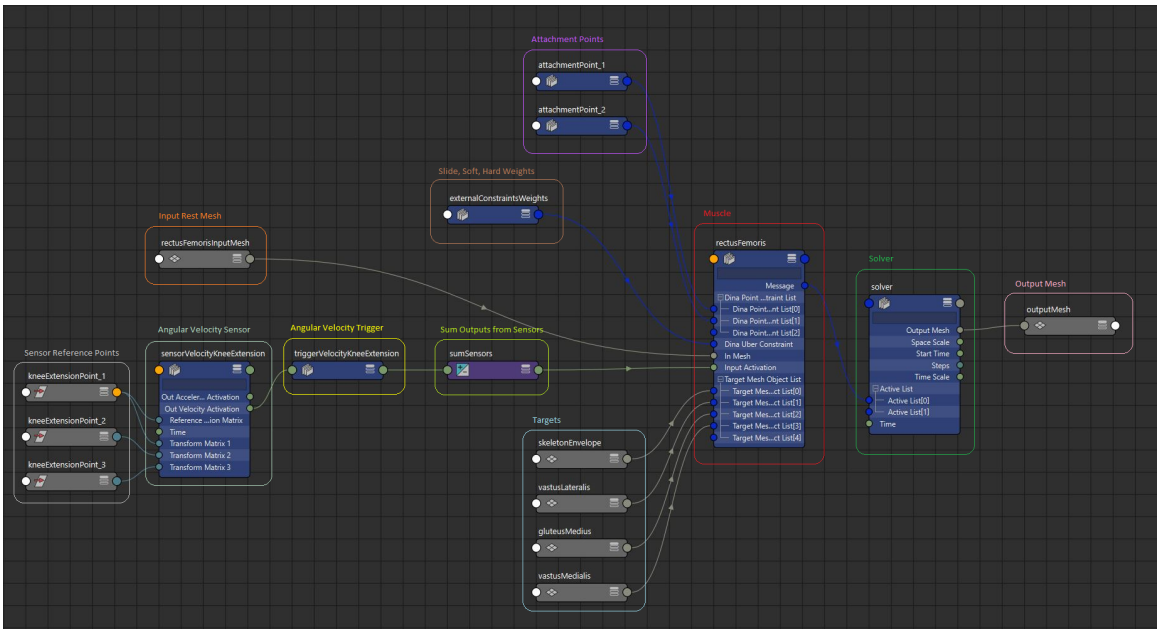

Figure 16: *A snapshot of the Autodesk Maya node editor interface. The graph describes the setup of the rectus femoris muscle produced by our plugin.*

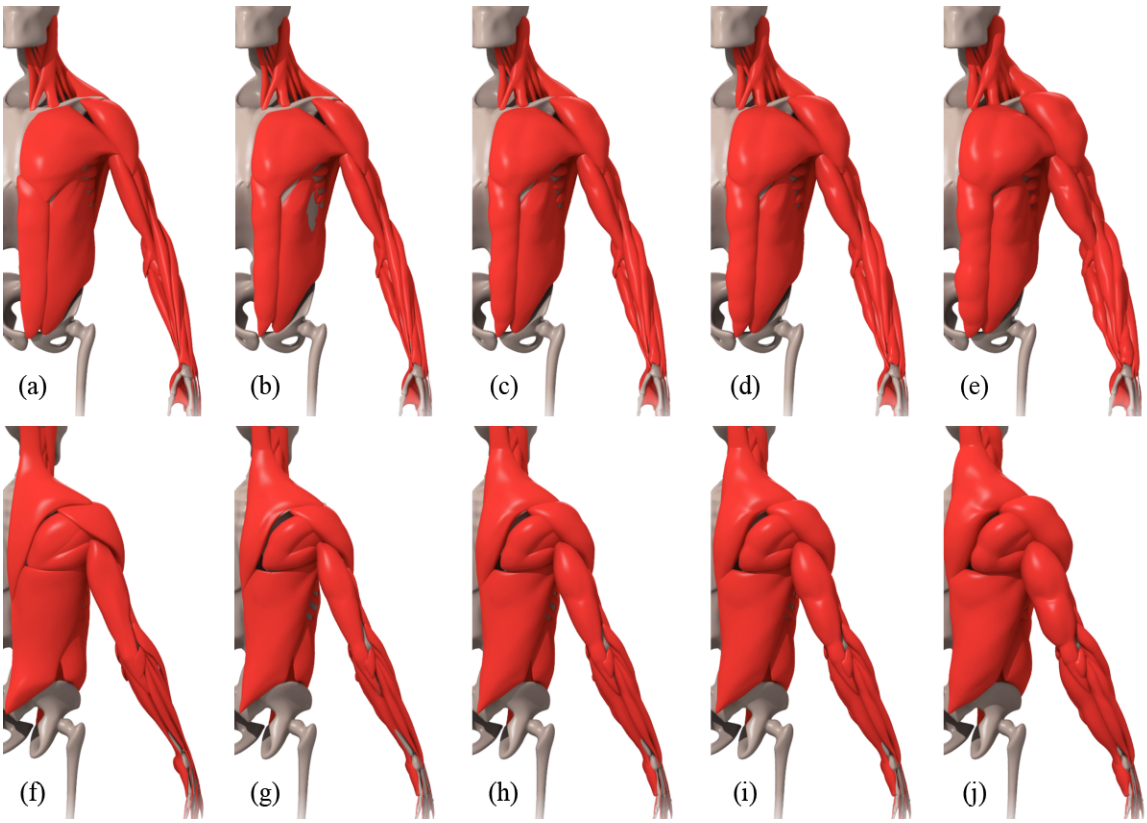

Figure 17: *The activation of muscle fibers is related to the growth in volume of the muscles using the overpressure system of Position Based Dynamics. The muscles at rest from front (a) and back (f), once activated can achieve different ratios of volume gain that can be controlled by the artist: ratio 1.0 or no gain (b and g); ratio 1.3 (c and h); ratio 1.5 (d and i); ratio 2.0 or double the volume (e and j).*

Figure 18: *Four frames from the simulation of running motion data seen from front and behind the character. The complete set of muscles is made of 140 separate meshes, coupled together by 135780 dynamic constraints. The result of the simulation of this fast motion did not produce any major intersection between muscles that would result in undesired artifacts on the skin of the character.*

References

[CBC15] COMER S., BUCK J., CRISWELL B.: Under the scalpel - ilm's digital flesh workflows. In *ACM SIGGRAPH 2015 Talks* (New York, NY, USA, 2015), SIGGRAPH '15, ACM, pp. 10:1–10:1. 1, 2

[Fra12] FRATARCANGELI M.: Position-based facial animation synthesis. *Computer Animation and Virtual Worlds 23*, 3-4 (2012), 457–466. 2

[GH24] GASSER H. S., HILL A. V.: The dynamics of muscular contraction. *Proceedings of the Royal Society of London B: Biological Sciences 96*, 678 (1924), 398–437. 2

[hou] Houdini fx 16.5. https://www.sidefx.com/products/houdini-fx/. Accessed: 2018-04-02. 2, 3

[JBE*16] JACOBS J., BARBIC J., EDWARDS E., DORAN C., VAN STRATEN A.: How to build a human: Practical physics-based character animation. In *Proceedings of the 2016 Symposium on Digital Production* (New York, NY, USA, 2016), DigiPro '16, ACM, pp. 7–9. 2

[Lan17] LANGLANDS A.: SIGGRAPH now: War for the planet of the apes. https://youtu.be/txoEDIdbUrg, 2017. Accessed: 2018-04-02. 2

[LGK*12] LEE D., GLUECK M., KHAN A., FIUME E., JACKSON K.: Modeling and simulation of skeletal muscle for computer graphics: A survey. *Found. Trends. Comput. Graph. Vis. 7*, 4 (Apr. 2012), 229–276. 2, 5

[may] Autodesk maya. https://www.autodesk.es/products/maya/overview. Accessed: 2018-04-03. 7

[MHHR07] MÜLLER M., HEIDELBERGER B., HENNIX M., RATCLIFF J.: Position based dynamics. *J. Vis. Comun. Image Represent. 18*, 2 (Apr. 2007), 109–118. 2, 3, 4, 6

[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* (New York, NY, USA, 2016), MIG '16, ACM, pp. 49–54. 1, 2, 4, 6

[MMCK14] MACKLIN M., MÜLLER M., CHENTANEZ N., KIM T.-Y.: Unified particle physics for real-time applications. *ACM Trans. Graph. 33*, 4 (July 2014), 153:1–153:12. 2

[MMT*16] MILNE A., MCLAUGHLIN M., TAMSTORF R., STOMAKHIN A., BURKARD N., COUNSELL M., CANAL J., KOMOROWSKI D., GOLDBERG E.: Flesh, flab, and fascia simulation on zootopia. In *ACM SIGGRAPH 2016 Talks* (New York, NY, USA, 2016), SIGGRAPH '16, ACM, pp. 34:1–34:2. 2

[NQMT91] NEERING I., QUESENBERRY L., MORRIS V., TAYLOR S.: Non uniform volume changes during muscle contraction. *The Journal of General Physiology 59*, 4 (1991), 926–933. 2, 4

[SLG14] STEELE T., LEPREVOST J. C., GOSSART K.: A position-based dynamics system for animated character effects. In *ACM SIGGRAPH 2014 Talks* (New York, NY, USA, 2014), SIGGRAPH '14, ACM, pp. 55:1–55:1. 2

[SSW*10] SEO J., SEOL Y., WI D., KIM Y., NOH J.: Rigging transfer. *Computer Animation and Virtual Worlds 21*, 3-4 (2010), 375–386. 2

[TBHF03] TERAN J., BLEMKER S., HING V. N. T., FEDKIW R.: Finite volume methods for the simulation of skeletal muscle. In *Proceedings of the 2003 ACM SIGGRAPH/Eurographics Symposium on Computer Animation* (Aire-la-Ville, Switzerland, Switzerland, 2003), SCA '03, Eurographics Association, pp. 68–74. 2

[tis] Tissue. https://www.wetafx.co.nz/research-and-tech/technology/tissue/. Accessed: 2018-04-02. 1, 2

[TRF16] TURCHET F., ROMEO M., FRYAZINOV O.: Physics-aided editing of simulation-ready muscles for visual effects. In *ACM SIGGRAPH 2016 Posters* (New York, NY, USA, 2016), SIGGRAPH '16, ACM, pp. 80:1–80:2. 2

[TSIF05] TERAN J., SIFAKIS E., IRVING G., FEDKIW R.: Robust quasistatic finite elements and flesh simulation. In *Proceedings of the 2005 ACM SIGGRAPH/Eurographics Symposium on Computer Animation* (New York, NY, USA, 2005), SCA '05, ACM, pp. 181–190. 2

[ziva] Ziva character platform. http://www.zivadynamics.com/character-platform-beta. Accessed: 2018-04-02. 2

[zivb] Ziva lion project. http://www.zivadynamics.com/lion-project. Accessed: 2018-04-02. 7

[zivc] Ziva VFX. http://www.zivadynamics.com/ziva-vfx. Accessed: 2018-04-02. 1

[ZM89] ZAJAC F. E., MICHAEL G. E.: Determining muscle's force and action in multi-articular movement. *Exercise and Sport Sciences Reviews 17*, 1 (Jan. 1989), 187–230. 2