



Fast and Realistic Approach to Virtual Muscle Deformation*

Martin Cervenka¹^a and Josef Kohout²^b

¹University of West Bohemia, Faculty of Applied Sciences,
Department of Computer Science and Engineering, Czech Republic

²University of West Bohemia, Faculty of Applied Sciences,
NTIS - New Technologies for the Information Society, Czech Republic

Keywords: Muscle Modelling, Muscle Deformation, Muscle Fibres, Position based Dynamics.

Abstract: This paper describes a real-time simulation of muscle movement that is based on inverse kinematics where bone placement is known a priori and muscle shape is calculated by a modified position-based dynamics (PBD) method. The method is comparable and competitive with the others, moreover, it is enhanced with some novel features like approach for respecting muscle anisotropy, really fast simplistic collision detection, etc. This real-time simulation presents visual plausibility of resulting muscle deformation in most cases.

1 INTRODUCTION

One of the diseases with high prevalence is osteoporosis (Wade et al., 2014). This disease causes weakening of bone tissue, which results in weak bones prone to break. This is a valid reason to be concerned about forces employed to softened bones. Bergmann et. al. (Bergmann et al., 2001) show that twice as much force is exerted during walking than during standing. In severe cases, there is a chance that a weak bone is not able to absorb surrounding forces and fractures.

Osteoarthritis is another musculoskeletal disease to consider. In advanced stages, a form of treatment is needed, which is usually done by replacing the bone joint with an artificial one. Knowledge of forces impacting the bone joint is essential for surgeons to choose suitable artificial joint (Oatis, 2013). An inaccurate choice of artificial joint may be painful for the patient or even cause further harm.


A well-built model can be useful to predict the discussed forces. Such a model should be physically correct and should be also patient-specific because body constitution varies in patients.


Modern technologies allow us to use computers to simulate the musculoskeletal system (or its model, respectively) and get some form of approximate values

from this process. There are some models (e.g. (Delp et al., 1990), (Arnold et al., 2009)) measured on single patient; unfortunately, we need patient-specific models. Even though it may seem like a good idea to use model measured on single patient for every other patient (for the sake of simplicity etc.), it neglects significant features of the individual patient. There are statistical models as well (e.g., PCA based statistical model using data from 26 patients by Zhang et al. (Zhang et al., 2016)), which can achieve better results in general, despite these statistical models cannot be as good as patient-specific models. However, getting complete patient-specific data is nearly impossible due to inaccurate measuring, complexity of data processing, time consumption etc.

To give a real-life example where the simulation is needed, calculation of the stresses to which bones are subjected when performing a specific action can provide fracture prediction for people suffering from osteoporosis. This information enables better prognosis and more precise treatment, and a dynamical analysis and visualisation of muscle activity may help to identify issues in the action of a professional athlete that can lead to more effective training procedures and the identification of ways in which performance can be improved. For such applications, a patient-specific or subject-specific musculoskeletal model is essential for the simulation and its visualisation.

We present a novel, real-time approach for muscle modelling, derived from position-based dynamics (well known in modern computer graphics field).

^a <https://orcid.org/0000-0001-9625-1872>

^b <https://orcid.org/0000-0002-3231-2573>

*This work was supported by the Ministry of Education, Youth and Sports of the Czech Republic, project SGS-2019-016 and project PUNTIS (LO1506).

PBD was originally proposed by Müller (Müller et al., 2007) and is currently still in development (e.g. cloth and fluid simulation (Shao et al., 2017)). Main contributions are:

- Using computer graphics related PBD (position based dynamic) method for muscle deformation,
- Extending PBD by concerning anisotropy of muscles,
- Fast minimalistic implementation of muscle modelling approach.

2 APPROACH

2.1 Static Model

Construction of a patient-specific model requires having data on the anatomy of the individual patient. Muscle parameters such as geometrical shape, attachment sites, fibre orientation, etc. are essential to get accurate results of simulations. The geometrical shape of a muscle can be extracted from MRI images and this can be done automatically, though with some issues. However, most parameters regarding the muscle architecture cannot be extracted from these images and generally, they are very difficult, if not impossible, to get *in vivo*. Nevertheless, this information can be obtained from cadaver measurements (not patient-specific). It is possible then to combine this general information with specific patient data.

In this paper, a subset of a comprehensive female cadaver anatomical dataset (81 y/o, 167 cm, 63kg) is used. Specifically, pelvic and femur bones together with several muscles from the pelvic region have been selected.

The complete data are publicly available in LHD dataset (Viceconti et al., 2008) and has been selected because it includes high-quality surface meshes of bones and muscles. Furthermore, the dataset was improved by removing non-manifold edges, duplicated vertices and degenerate triangles followed by surface smoothing in both muscle and bone models using MeshLab (Cignoni et al., 2008). The dataset also contains muscle attachment areas and geometrical paths of superficial fibres obtained from dissection (Van Sint Jan, 2005).

2.2 Dynamic Model

Having a static model, a dynamic one can be obtained quite simply providing that the motion data for that model are available. If bones are assumed

as completely rigid, motion can be simulated via inverse kinematics. Inverse kinematics means that the location and movement of all bones are known, and muscle actual shape has to be determined according to these situations. We note this is exactly the opposite to what can be seen in real situation, where muscles control bone movement.

In this paper, position-based dynamics (PBD), which was introduced in (Müller et al., 2007) as a fast, stable, and controllable solution to various problems in computer graphics, e.g., simulations of cloth or fluids, is used to reposition the vertices of the surface mesh of muscle with the modelling constraints to preserve the shape and volume of the muscle and prevent mutual penetration of the muscle with the bones. Details will be described in Section 3.

The muscle is then decomposed into a set of fibres using either Kukacka (Kohout and Kukacka, 2014) or CHMD (Kohout and Cholt, 2017) method, depending on which one is more appropriate in the concrete case. Note that this set represents the dynamics of muscle architecture. Details regarding muscle decomposition will be described in Section 5.5.

3 PBD

Müller (Müller et al., 2007) in his paper described PBD with four requirements:

1. Preserve local distances,
2. Maintain local shape,
3. Keep the original volume,
4. Not collide with other models.

In the context of muscle modelling, another requirement, which considers muscle anisotropy, must be introduced to model muscle motion accurately. This will be described later.

PBD method is based on solving equation 1 for discretized muscle model over and over again. It describes a movement of a single point (vertex) during simulation ($\Delta \mathbf{p}_i$ denote the difference in position of i th point of the model), maintaining various features. Mentioned features can be for example volume or shape preservation, but it can be anything giving meaning. Some of the preserved features are mentioned and further described later in the text.

$$\Delta \mathbf{p}_i = - \frac{\nabla_{p_i} C(\mathbf{p}_1, \dots, \mathbf{p}_n)}{\sum_j |\nabla_{p_j} C(\mathbf{p}_1, \dots, \mathbf{p}_n)|^2} \cdot C(\mathbf{p}_1, \dots, \mathbf{p}_n) \quad (1)$$

In above equation, C is a constraint function (more details below), n is the number of points in the muscle model and j is the number of constraint functions.

Mathematically speaking, constraint function and its gradients must be known to solve the deformation problem.

3.1 Distance Constraint

A rather basic constraint, which we can imagine, is restricting each model point to change the distance from the others in its neighbourhood. This constraint can be formulated as is in (2), where d is the original distance between points \mathbf{p}_1 and \mathbf{p}_2 .

$$C(\mathbf{p}_1, \mathbf{p}_2) = |\mathbf{p}_1 - \mathbf{p}_2|_2 - d \quad (2)$$

3.2 Volume Constraint

Next, we can restrict the muscle model to change its volume during the simulation process. To calculate this constraint, we need to know how to measure the volume of the model. Assuming triangular mesh, the volume can be computed tetrahedron by tetrahedron, which in summation eventually leads to constraint function:

$$C(\mathbf{p}_1, \dots, \mathbf{p}_n) = \sum_{i=1}^m \left(\mathbf{p}_{i_1}^i \cdot (\mathbf{p}_{i_2}^i \times \mathbf{p}_{i_3}^i) \right) - V_0 \quad (3)$$

In this function, m describes number of triangles forming the muscle model, V_0 is muscle original volume and p_{i_j} denotes j th point of the i th triangle.

3.3 Local Shape Constraint

Above described constraints are not enough to prevent the surface from becoming noisy, full of unrealistic spikes. One possible solution to this problem is to use the distance constraint not only to keep the distances between adjacent points but also between the pairs of points lying on the opposite sides of the muscle. This would, however, need to create a 3D mesh first, which would be quite complex to do. Another option is to ensure that the local shape is maintained. To achieve this, the angles between neighbouring triangles should stay the same during deformation. Calculating the angle of two triangles is integrated into equation (4), where two triangles are described by four points, sharing points with index 1 and 2.

$$\begin{aligned} C(\mathbf{p}_1, \mathbf{p}_2, \mathbf{p}_3, \mathbf{p}_4) &= \arccos(\mathbf{n}_1 \cdot \mathbf{n}_2) - \varphi_0 \\ &= \arccos \left(\frac{(\mathbf{p}_2 - \mathbf{p}_1) \times (\mathbf{p}_3 - \mathbf{p}_1)}{|\mathbf{p}_2 - \mathbf{p}_1|_2 \times |\mathbf{p}_3 - \mathbf{p}_1|_2} \right) \cdot \\ &\quad \cdot \frac{(\mathbf{p}_2 - \mathbf{p}_1) \times (\mathbf{p}_4 - \mathbf{p}_1)}{|\mathbf{p}_2 - \mathbf{p}_1|_2 \times |\mathbf{p}_4 - \mathbf{p}_1|_2} - \varphi_0 \end{aligned} \quad (4)$$

3.4 Anisotropy

The PBD algorithm has been originally proposed in the computer graphics field to model isotropic materials (e.g., cloths). However, muscles are anisotropic (may behave differently in two distinct directions), so it is appropriate to take anisotropy into account. The main idea is that muscle surface is stiffer in the direction perpendicular to the muscle fibres and more flexible in the direction parallel to these fibres. Mathematically speaking, we multiply the distance constraint in (2) with the result of equation (5).

$$k_i = 1 - \mathbf{u}_i \cdot \mathbf{v}_i \quad (5)$$

The direction of i th edge is described by normalized vector \mathbf{u}_i , \mathbf{v}_i denotes tangential direction normal vector of nearest fibre on the surface. If both vectors are collinear, the result k_i will be zero, meaning no distance is preserved. If these two vectors are perpendicular, then k_i is equal to one and edge length will be well preserved. This behaviour is assured because value k_i multiplies the result of distance constraint function in (2).

4 COLLISION HANDLING

In the simulation, moving muscles and bones should not intersect each other. There are some methods with different advantages and disadvantages, some of them are described below.

4.1 Brute Force

The most simple way to detect a collision is to test each primitive of a surface mesh with all primitives of the other mesh for an intersection. Time complexity in big-O notation is $O(n^2)$, where n is the number of primitives (triangles in our case), which makes the brute force approach suitable only for small models, for which the overhead associated with more sophisticated approaches is not amortized. For bigger (real) models, we need a sort of space division algorithm.

4.2 Bounding Volume Hierarchies

One of the data structure suitable for space division is a bounding volume hierarchy. The basic idea behind the structure is that space is recursively subdivided employing some volumetric (in 3D, planar in 2D) geometrical primitives (box, sphere, etc.). In big-O notation, we achieve $O(n \log n)$ or even $O(\log^2 n)$, assuming the number of vertices of both tested meshes

depends linearly on each other. The main disadvantage is that the structure has to be built before the simulation or even updated every time the muscle shape changes.

4.2.1 octree

A special case of bounding volume hierarchies is an octree. The octree divides the three-dimensional box into eight (therefore *octree*) smaller boxes (box is divided in half in each axis). The division is done recursively and stops when the box contains a sufficiently small number of elements, i.e., vertices or triangles.

4.3 Voxelization

In this approach, the given three-dimensional box is divided into n^3 (n division in each axis) equally sized boxes. If n is selected carefully, every box contains a (sufficiently small) constant number of element, so testing can be theoretically done in $O(1)$ time using big-O notation, however, memory complexity will be $O(n^3)$ as far as every cube content has to be stored in memory.

Our goal is to do a fast deformation, so we decided memory is not an issue, therefore voxelization is used in the proposed approach.

4.4 Collision Response

In our case, two main scenarios have to be distinguished. In the first one, a muscle moves (e.g. because of surrounding forces) and hits a bone. When vertices of the muscle collide, they are pushed back in the direction they enter the bone so they will not penetrate the bone anymore.

In the second scenario, a bone moves into a muscle. In this case, the colliding muscle vertices did not "enter" the bone, so we do not know the entering direction. We propose a solution where the same transformation used on colliding bone is applied to colliding muscle vertices as well.

5 EXPERIMENTAL RESULTS

The real data described in Section 2.1 is used along with an artificial dataset (described below) to test the proposed approach. The results of the experiments are described in detail below.

5.1 Artificial Data

To test collision detection and behaviour in extreme cases, we prepared an artificial dataset, where a box of 5292 triangles on its surface is squished between two plates (12 triangles each). The initial setup is visualized in Fig.1.

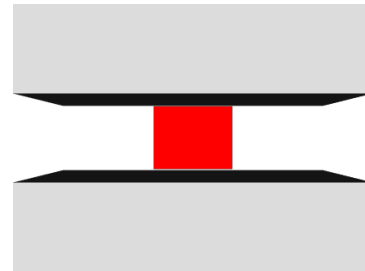


Figure 1: Input artificial data.

At first, in one hundred iterations, the top plate moves to decrease the space between both plates. The distance between plates in 100th iteration is 10% of the original distance in the first iteration. Inverse movement is then performed between 100th and 200th iteration, returning the plates to their initial position. Additional one hundred iterations are used for box stabilization.

As it can be seen in Fig.2, the box is squished quite a lot. Despite the fact, it returns to its original shape in 300th iteration (only with slight rotation caused by asymmetrical triangulation). Even in 200th iteration the results is acceptable, except one corner of the box.

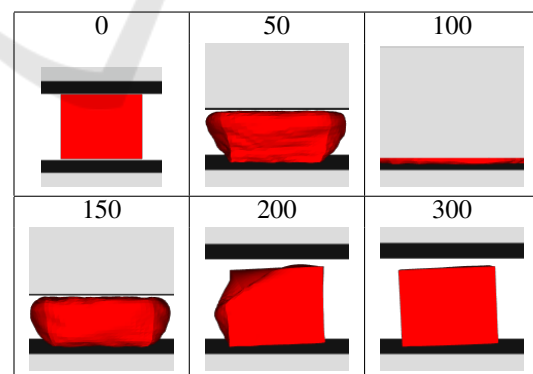


Figure 2: Results in different simulation frames (artificial data).

5.2 Iliacus

We also tested iliacus muscle, which connects the femur and pelvic bones from the front side. The bones and the muscle are visualized in Fig.3. The muscle

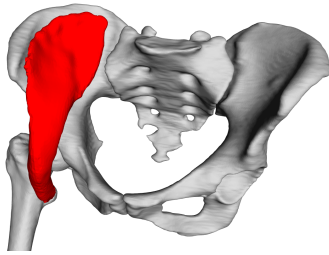


Figure 3: Input iliacus muscle and adjacent bones.

and bones are closed triangular meshes with 13858 and 254442 (42456 for femur) triangles, respectively.

A very similar simulation scenario like in the artificial data case has been applied to iliacus dataset. The first hundred iterations are used for hip flexion. The femur bone rotates one radian during this movement. The second hundred iteration is allocated for the inverse movement. The last hundred (200^{th} - 300^{th}) iterations are present to stabilize the muscle.

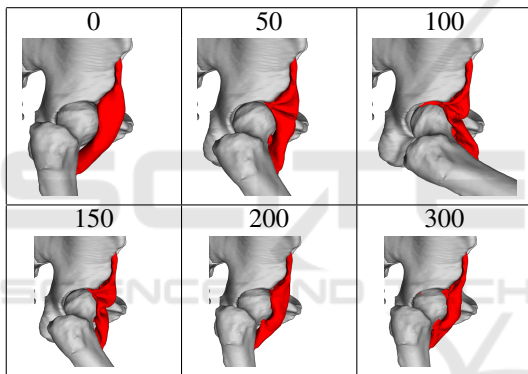


Figure 4: Results in different simulation frames (Iliacus muscle data).

The posterior part of the iliacus muscle is pushed into the joint during the flexion, as it can be seen in Fig.4. The explanation for this behaviour is as follows. This part of the mesh is unrealistically arched towards the joint and, therefore, distance and local shape constraints tend to move the points of this part closer to the joint. As the femur and pelvic bones do not touch, there is a narrow space into which this part of the mesh can squeeze, and from which it is difficult to get out. Even though the result is not visually plausible, the quantitative tests (described later) show that all key factors of the muscle are preserved as much as possible.

5.3 Gluteus Maximus

Gluteus maximus muscle is attached to the same bones as iliacus muscle but from the other side. Tri-



Figure 5: Input gluteus maximus muscle and adjacent bones.

angular mesh of the consists of 19752 triangles. Fig.5 shows how the model looks like.

The muscle undergoes the same movement scenario as iliacus mentioned above. Visualization in 6 important iterations is shown in Fig.6. As we can see, the result in iteration 300 is nearly the same as in the first iteration (the original muscle pose).

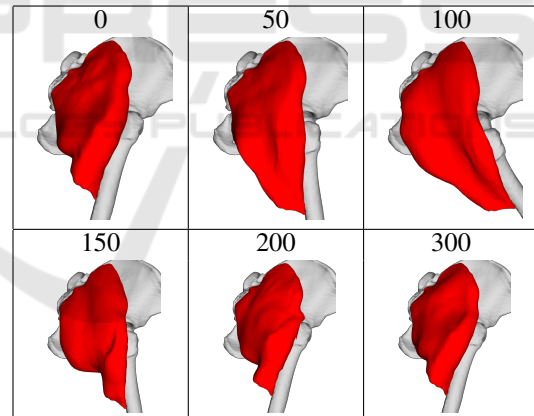


Figure 6: Results in different simulation frames (gluteus maximus muscle data).

5.4 Other Muscles

Within this testing procedure, we have tested gluteus medius and adductor brevis muscles. Both muscles deform quite realistically, as it can be see in Fig.7 and Fig.8, where the situation in the maximal hip flexion (same scenario like in gluteus maximus and iliacus case) is shown.

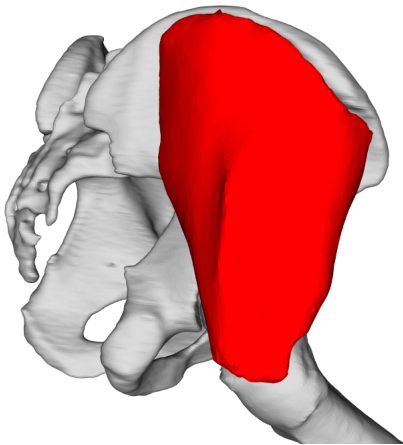


Figure 7: Gluteus medius in maximum flexion position.

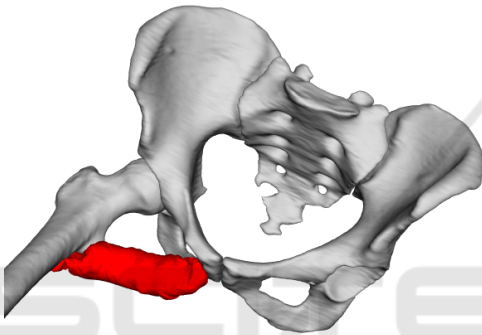


Figure 8: Adductor brevis in maximum flexion position.

5.5 Muscle Decomposition

The approach described so far works primarily with the surface model of a muscle. To calculate the forces (and other physical properties of the muscle), the muscle needs to be decomposed into individual fibres. This can be done using, for example, Kukacka or CHMD muscle decomposition methods, which were proposed in (Kohout and Kukacka, 2014) and (Kohout and Cholt, 2017), respectively. In the following subsections, we briefly describes these methods.

5.5.1 Kukacka Decomposition

The inputs of the Kukacka decomposition method (Kohout and Kukacka, 2014) are:

- Triangular (and manifold) surface model of the muscle to decompose,
- Fibre template, giving information about internal fibre arrangement,
- Attachment areas to adjacent bones (origin and insertion), defined by a set of points lying on the adjacent bone surface models

Decomposition is then done as follows. Attachment areas are projected from the bone surface onto the muscle surface to define the parts of the muscle that are subsequently removed. Isocontours are then computed on the modified muscle model, using a piece-wise linear scalar field. The scalar field has its maximum on the insertion boundary vertices, whereas it has its minimum on the origin boundary vertices. User can specify how many isocontours are generated in this process.

Similarly to (Delp, 2005), muscle fibre architecture (template) is represented by a unit 3D space with arbitrary (user-defined) number of fibres inside the space. The fibres are represented analytically by Bezier spline curves. From multiple templates, one is selected according to the muscle being modelled (depends on if it has parallel or pennate fibres, optionally on a pennate angle, etc.) and it is mapped one-to-one on isocontours calculated in the previous step, forming the fibres going through the muscle model. Finally, noise is eliminated using quadratic smoothing to make the result more realistic and visually plausible.

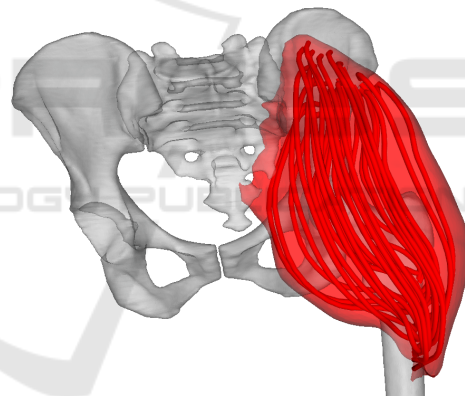


Figure 9: Gluteus maximus decomposed to individual fibres by Kukacka's (Kohout and Kukacka, 2014) algorithm.

5.5.2 CHMD Decomposition

A technique by Kohout & Cholt (Kohout and Cholt, 2017) performs a centripetal Catmull-Rom interpolation of the input fibres lying on the surface model of the muscle, or nearby, to get the fibres inside the muscle. Their approach can even work with multiple headed muscles, distributing the fibres automatically among the heads.

In comparison with Kukacka's proposed method, this method needs specification of fibres on the surface, which typically requires some manual effort because these are not available for the patient but must be adopted from a cadaveric dataset. On the

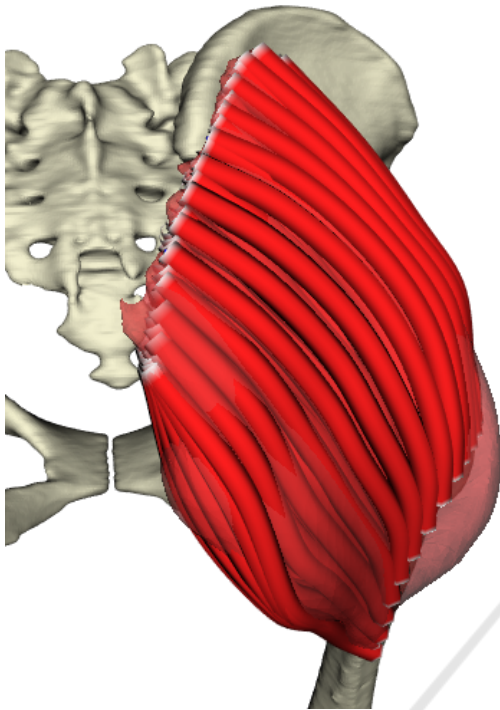


Figure 10: Gluteus maximus decomposed to fibres by Cholt's (Kohout and Cholt, 2017) algorithm.

other hand, it can work with multiple headed muscles, whereas Kukacka's approach can not.

5.6 Quantitative Tests

To make some exact result, we use quantitative tests. These tell us how well are constraints satisfied during simulation.

Volume preservation constraint is tested by determining ratio between both original and actual volumes. Fig.11 for artificial data, Fig.12 and Fig.13 for real data respectively, show us the volume preservation results.

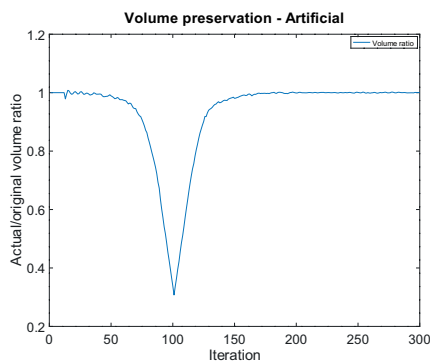


Figure 11: Volume preservation of artificial data.

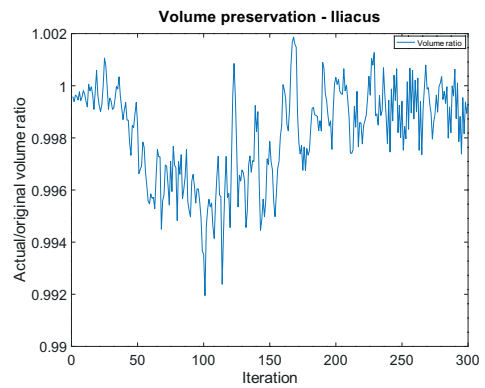


Figure 12: Volume preservation of iliacus muscle data.

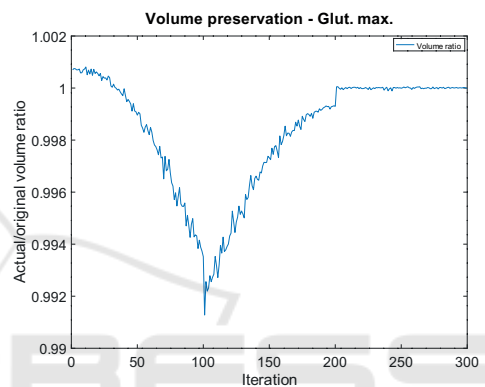


Figure 13: Volume preservation of gluteus maximus muscle data.

As we can see from the results, the volume is well preserved in both real data (the error is less than 1% in both cases), on artificial data, there is a nice curve describing squishing (reducing) box volume during the simulation and then restoring.

Next measurable property is average edge extension. At first, we cannot say much about artificial data (squished box) from the plot on Fig.14. As for the iliacus muscle, some edges remain longer than normal (see Fig.15) because they are stuck in the hip joint. In the case of gluteus maximus dataset (Fig.16), the first 100 iterations show edge extension during hip flexion (it is correct behaviour because muscle extends in this phase) and the second hundred iterations return the average length extension to almost 1 (i.e., the muscle returns to its original pose correctly). We can also see that the last 100 iterations are not crucial in this scenario.

We also tested how well the dihedral angles are preserved during the simulation. In this paper, the dihedral angle is the angle between two adjacent triangles in the muscle triangle mesh. According to plots in Fig.17, Fig.18 and Fig.19, we can conclude that there are some pairs of triangles which do not pre-

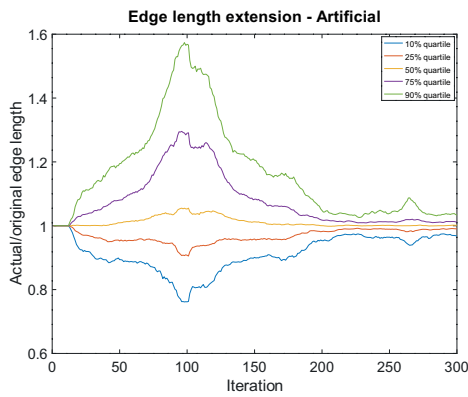


Figure 14: Average edge extension of artificial data.

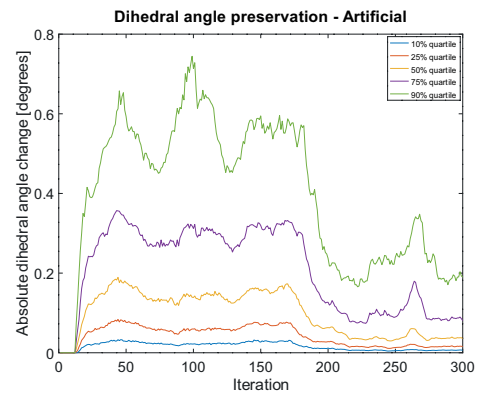


Figure 17: Average absolute dihedral angle change of artificial data.

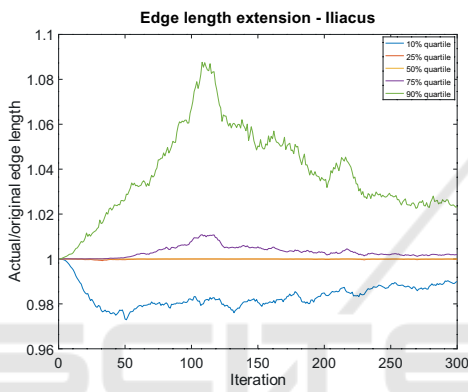


Figure 15: Average edge extension of iliacus muscle data.

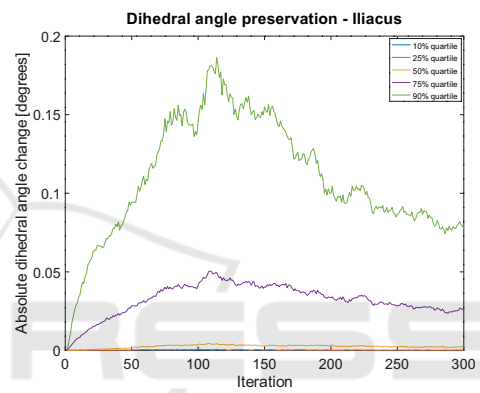


Figure 18: Average absolute dihedral angle change of iliacus muscle data.

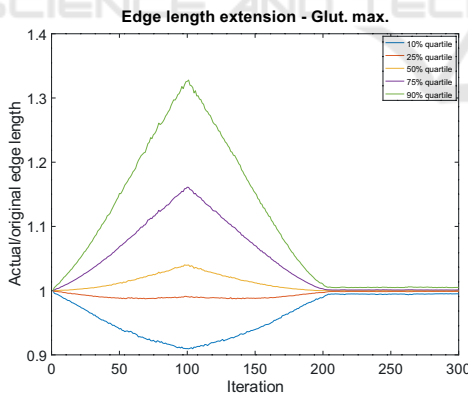


Figure 16: Average edge extension of gluteus maximus muscle data.

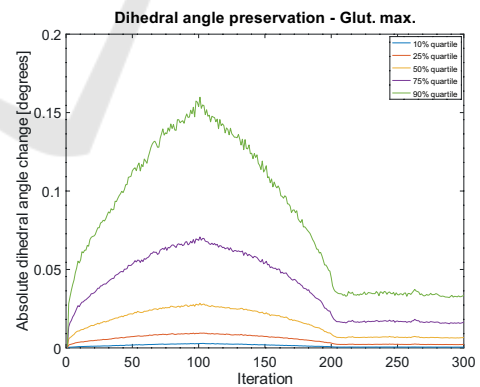


Figure 19: Average absolute dihedral angle change of gluteus maximus muscle data.

serve its original angle, but most of them do.

5.7 Fibre Length

Last but not least, the lengths of fibres were analyzed. In the iliacus muscle case, as we can see in Fig.20, many length curves exhibit two big bumps shortly after 100th iteration. This is caused by the fact that a

part of the muscle is stuck in the hip joint (as discussed previously). Nevertheless, when the bones return to their initial rest-pose (i.e., after 200 iterations), the vast majority of fibres restore their original lengths quite well. Gluteus maximus muscle behaves as expected – see Fig.21. During the flexion, all lengths increase; during the extension, they decrease.

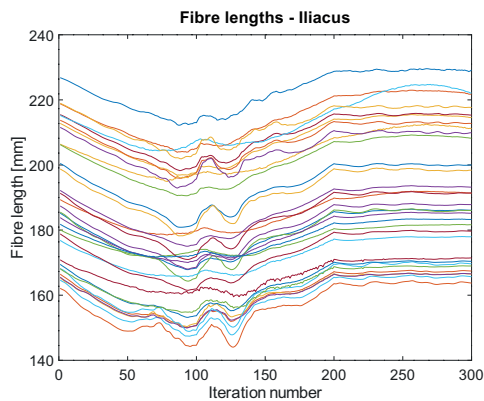


Figure 20: Total length of each individual fibre during simulation in iliacus muscle.

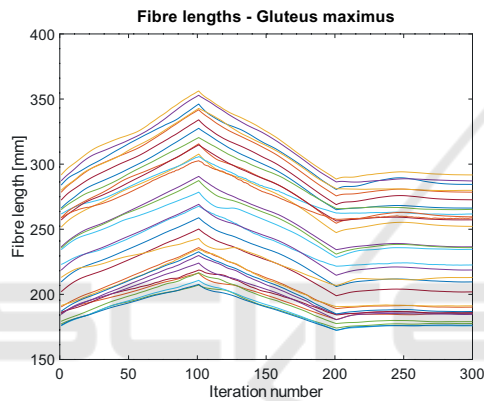


Figure 21: Total length of each individual fibre during simulation in gluteus maximus muscle.

5.8 Speed

The proposed method was designed to be not only precise, but mainly, fast. It was implemented in C++ using VTK toolkit. Its current version is publicly available at <https://github.com/cervenkam/muscle-deformation-PBD>.

All testing scenarios above were measured how fast each one is. FPS (Frames Per Second) is used as speed metric in this case.

All tests were performed on Intel® Core™ i7-4930K 3.40GHz CPU, Radeon HD 8740 GPU and WDC WD40EURX-64WRWY0 4TB HDD. Results are listed on Tab.1.

As it can be seen from the results, FPS strictly depends on number of triangles (Spearman's $\rho = -1$). The more triangles is used, the slower the method is.

Even though the program is mostly unoptimized and runs sequentially at the moment, the FPS is sufficient for considered purposes in general.

Table 1: FPS of each simulation.

Deforming object	Triangle count	FPS
Gluteus maximus	19752	33.85
Abductor brevis	17124	35.89
Iliacus	13858	47.21
Gluteus medius	10622	57.12
Artificial box	5292	153.61

6 DISCUSSION

The most simple approach to muscle deformation problem is probably to use line segments to approximate both muscles and tendons. An example is shown in Fig.22. The coordinates of each end-point of the corresponding line segments are updated when bone moves, causing shortening or extending of the line segments. Various values (e.g., moment arms) can be consequently calculated from the length of each line. These models are popular in practice (they are used, e.g., in AnyBody¹ or OpenSim²) because of their simplicity and speed. However, the accuracy of calculations based on these models is, especially, for complex muscles, e.g., gluteus medius, low (Delp, 2005). A model can be more precise if we assume more lines (Valente et al., 2012) per a muscle or if we use more complex lines wrapping around some kind of parametric surfaces (e.g. spheres, cylinders (Audenaert and Audenaert, 2008), etc.), nonetheless, all of these are difficult to set up. This may be the reason why most studies use gait2392 model shipped with OpenSim software even though there are no more than three lines per muscle and these lines penetrate the bones in some poses. Using our approach, described in this paper, the user can easily generate hundreds of lines (i.e., fibres) automatically and impenetrability of muscles and bones is guaranteed.

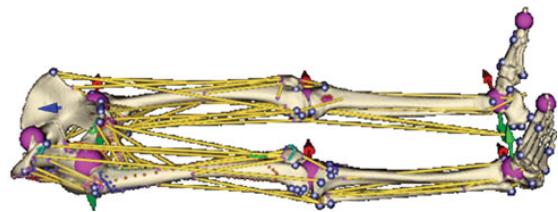


Figure 22: Muscle approximation using line segments – yellow lines (taken from (Kohout et al., 2013)).

Position-based dynamics (PBD), which is the core part of our approach, was firstly introduced in (Müller et al., 2007) as a computer graphics algorithm. Since

¹<https://www.anybodytex.com/software/>

²<https://opensim.stanford.edu/>

then, it has been further developed (e.g., (Macklin et al., 2019) proposed recently some speed and accuracy improvements) and has found many (close to) real-time applications, not only in computer graphics but even in other domains. For example, Kotsalos et al. use PBD to model blood cells (Kotsalos et al., 2019). As far as we know, however, there is no PBD-based method for muscle modelling even though one could expect a good compromise between speed and accuracy from such a method.

A mass-spring system (MSS) is another approach to consider. Janak et al. use MSS to approximate muscle (Janak, 2012), showing promising, simple method with visually plausible results. However, there are some issues in the approach they proposed. First, to avoid penetration between muscles and bones, the authors choose a particle-based collision detection method requiring many particles to get reasonable results, which, however, causes high time and memory complexity. Secondly, and more importantly, the main issue is that muscle volume is not preserved during deformation. This could be probably solved using the approach described in (Hong et al., 2006), however, it would increase computational time dramatically. Finally, our experiments show that although this method retains the smooth shape of iliacus muscle during flexion, it twists the part of the muscle close to the insertion. This is because, unlike our method, the particles are in the entire volume of the muscle, which results in a model that is much more rigid, and as anisotropy is not exploited, rigid in all directions. Our method supports anisotropy, preserves the volume and runs in a fraction of time while requiring no extra parameter or input in comparison with this method.

On the contrary to line segment approximation, finite element method (FEM) is the most complex method. Well discretized muscle provides a physically very accurate result (see e.g., (Delp, 2005)). However, computational complexity is high, meaning the FEM-based methods are unsatisfactorily slow. Therefore, it is quite impractical for real-time application or even clinical assessments. Next issue is a difficult set up of FEM methods, making them unsuitable for personalised musculoskeletal method deformation. Despite these facts, these methods can be seen in the movie industry, see e.g. Ziva VFX³ plugin for Maya, and in muscle physiology research, see e.g. (Oberhofer et al., 2009) or (Kojic et al., 1998). In comparison with these methods, our method is quite simple to set up and runs fast providing the promising results in most cases.

³<https://zivadynamics.com/>

7 CONCLUSION & FUTURE WORK

The proposed muscle deformation technique is capable to do fast and relatively accurate simulation. Despite problems with muscle trapped in the hip joint, we believe that a better collision detection can fix the issue.

Moreover, the method is ready to be included in OpenSim (a state-of-the-art simulation software) as a plugin, allowing common users to use the method more intuitively. Its source code is available at <https://github.com/cervenkam/muscle-deformation-PBD>.

In this paper, we verified the method, but to prove correctness, the method needs to be validated in real life. There are some works (e.g. (Modenese et al., 2018)) providing correct momentum values during muscle movement, which can be useful for validation.

ACKNOWLEDGMENT

Authors would like to thank their colleagues and students for valuable discussion, worthwhile suggestions and constructive comments. Authors would like to thank also anonymous reviewers for their hints and notes provided.

REFERENCES

- Arnold, E., Ward, S., Lieber, R., and Delp, S. (2009). A model of the lower limb for analysis of human movement. *Annals of biomedical engineering*, 38:269–79.
- Audenaert, A. and Audenaert, E. (2008). Global optimization method for combined spherical-cylindrical wrapping in musculoskeletal upper limb modelling. *Computer methods and programs in biomedicine*, 92:8–19.
- Bergmann, G., Deuretzbacher, G., Heller, M., Graichen, F., Rohlmann, A., Strauss, J., and Duda, G. (2001). Hip contact forces and gait patterns from routine activities. *Journal of Biomechanics*, 34(7):859 – 871.
- Cignoni, P., Callieri, M., Corsini, M., Dellepiane, M., Ganovelli, F., and Ranzuglia, G. (2008). Meshlab: an open-source mesh processing tool. *Computing*, 1:129–136.
- Delp, S. (2005). Three-dimensional representation of complex muscle architectures and geometries 1. *Annals of Biomedical Engineering - ANN BIOMED ENG*, 33:1134–1134.
- Delp, S. L., Loan, J. P., Hoy, M. G., Zajac, F. E., Topp, E. L., and Rosen, J. M. (1990). An interactive graphics-based model of the lower extremity to study orthopaedic surgical procedures. *IEEE Transactions on Biomedical Engineering*, 37(8):757–767.

- Hong, M., Jung, S., Choi, M.-H., and Welch, S. (2006). Fast volume preservation for a mass-spring system. *IEEE computer graphics and applications*, 26:83–91.
- Janak, T. (2012). Fast soft-body models for musculoskeletal modelling. Technical Report DCSE/TR-2012-5, University of West Bohemia, Faculty of Applied Sciences.
- Kohout, J. and Cholt, D. (2017). Automatic reconstruction of the muscle architecture from the superficial layer fibres data. *Computer Methods and Programs in Biomedicine*, 150.
- Kohout, J., Clapworthy, G., Zhao, Y., Tao, Y., Gonzalez-Garcia, G., Dong, F., Wei, H., and Kohoutová, E. (2013). Patient-specific fibre-based models of muscle wrapping. *Interface focus*, 3:20120062.
- Kohout, J. and Kukacka, M. (2014). Real-time modelling of fibrous muscle. *Computer Graphics Forum*, 33(8):1–15.
- Kojic, M., Mijailovic, S., and Zdravkovic, N. (1998). Modelling of muscle behaviour by the finite element method using hill's three-element model. *International Journal for Numerical Methods in Engineering*, 43(5):941–953.
- Kotsalos, C., Latt, J., and Chopard, B. (2019). Bridging the computational gap between mesoscopic and continuum modeling of red blood cells for fully resolved blood flow. *Journal of Computational Physics*, 398. cited By 0.
- Macklin, M., Storey, K., Lu, M., Terdiman, P., Chentanez, N., Jeschke, S., and Müller, M. (2019). Small steps in physics simulation. In *Proceedings of the 18th Annual ACM SIGGRAPH/Eurographics Symposium on Computer Animation*, SCA '19, pages 2:1–2:7. New York, NY, USA. ACM.
- Modenese, L., Montefiori, E., Wang, A., Wesarg, S., Viceconti, M., and mazzà, C. (2018). Investigation of the dependence of joint contact forces on musculotendon parameters using a codified workflow for image-based modelling. *Journal of Biomechanics*.
- Müller, M., Heidelberger, B., Hennix, M., and Ratcliff, J. (2007). Position based dynamics. *Journal of Visual Communication and Image Representation*, 18:109–118.
- Oatis, C. (2013). *Kinesiology: The mechanics and pathomechanics of human movement: Second edition*. Lippincott Williams & Wilkins.
- Oberhofer, K., Mithraratne, K., Stott, N. S., and Anderson, I. A. (2009). Anatomically-based musculoskeletal modeling: prediction and validation of muscle deformation during walking. *Vis. Comput.*, 25(9):843–851.
- Shao, X., Liao, E., and Zhang, F. (2017). Improving sph fluid simulation using position based dynamics. *IEEE Access*, PP:1–1.
- Valente, G., Martelli, S., Taddei, F., Farinella, G., and Viceconti, M. (2012). Muscle discretization affects the loading transferred to bones in lower-limb musculoskeletal models. *Proc. Inst. Mech. Eng. H J. Eng. Med.*, 226(2):161–169.
- Van Sint Jan, S. (2005). Introducing anatomical and physiological accuracy in computerized anthropometry for increasing the clinical usefulness of modeling systems. *Critical Reviews in Physical and Rehabilitation Medicine*, 17:149–174.
- Viceconti, M., Clapworthy, G., and Van Sint Jan, S. (2008). The virtual physiological human — a european initiative for in silico human modelling —. *The journal of physiological sciences : JPS*, 58:441–6.
- Wade, S. W., Strader, C., Fitzpatrick, L. A., Anthony, M. S., and O'Malley, C. D. (2014). Estimating prevalence of osteoporosis: examples from industrialized countries. *Archives of Osteoporosis*, 9(1):182.
- Zhang, J., Fernandez, J., Hislop-Jambrich, J., and Besier, T. F. (2016). Lower limb estimation from sparse landmarks using an articulated shape model. *Journal of Biomechanics*, 49(16):3875 – 3881.