Muscle Deformation Using Position Based Dynamics

Josef Kohout[1(✉)] and Martin Červenka[2]

[1] NTIS - New Technologies for the Information Society, Faculty of Applied Sciences, University of West Bohemia, Univerzitní 8, Plzeň, Czech Republic
besoft@ntis.zcu.cz

[2] Department of Computer Science and Engineering, Faculty of Applied Sciences, University of West Bohemia, Univerzitní 8, Plzeň, Czech Republic
cervemar@kiv.zcu.cz

Abstract. This paper describes an approach to personalized musculoskeletal modelling, in which the muscle represented by its triangular mesh is subject to deformation, based on a modified position-based dynamic (PBD) method, followed by decomposition of its volume into a set of muscle fibres. The PBD was enhanced by respecting some muscle-specific features, mainly its anisotropy. The proposed method builds no internal structures and works only with the muscle surface model. It runs in real-time on commodity hardware while maintaining visual plausibility of the resulting deformation. For decomposition, the state-of-the-art Kukačka method is used. Experiments with the gluteus maximus, gluteus medius, iliacus and adductor brevis deforming during the simulation of the hip flexion and decomposed into 100 fibres of 15 line segments show that the approach is capable of achieving promising results comparable with those in the literature, at least in the term of muscle fibre lengths.

Keywords: Position based dynamics · Musculoskeletal system · Muscle deformation · Muscle fibres · Personalised model

1 Introduction

For decades, musculoskeletal modelling has been an important topic of research interest because of its ability to estimate internal loading on the human skeleton, which cannot be measured *in-vivo*. These estimations are useful, e.g., for preoperative surgical planning and postoperative assessment in orthopaedic surgery, rehabilitation procedures, prosthesis design, or prevention of injuries in professional sport.

Musculoskeletal models used in common practice (see, e.g., [1,2,6,8,11]) represent a muscle (or even a group of muscles) as one or more Hill-type one-dimensional structures, commonly referred as lines of action or fibres, connecting

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

X. Ye et al. (Eds.): BIOSTEC 2020, CCIS 1400, pp. 486–509, 2021.
https://doi.org/10.1007/978-3-030-72379-8_24

the origin and insertion points of the muscle, i.e., the sites at which the muscle is attached to the bone by a tendon, and passing through a couple of predefined via points, fixed to the underlying bone, or wrapping around predefined parametric objects (e.g. spheres, cylinders, or ellipsoids). Due to apparent difficulties with the specification of the locations of insertion, origin, and via points, it is common that there are no more than three fibres per muscle and they penetrate the bones in some poses. Figure 1 shows an example of models of this kind. An advantage of this approach, which makes it so popular, is its simplicity and rapid processing speed.

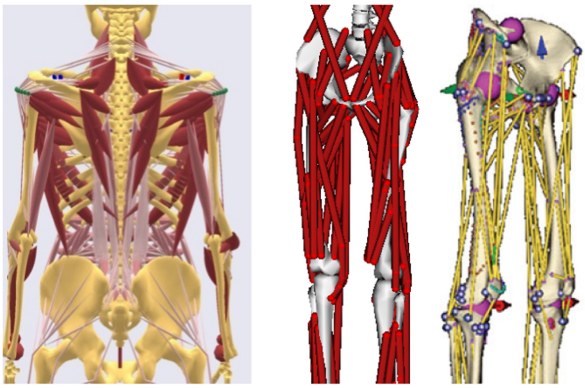

Fig. 1. Musculoskeletal models used in common practice: left – Anybody (http://www.anybodytech.com/) default model, middle – OpenSim (https://simtk.org/home/opensim) gait2392 model, right – LHDL model [22].

As acquiring complete patient-specific or subject-specific data is nearly impossible due to technological limitations of scanning devices, these musculoskeletal models have anatomical parameters derived from cadaver experiments. However, to answer specific subject-related questions, it is generally believed that a patient-specific or subject-specific model is needed. The current practice is, therefore, to take some of these generic models and adapt it to get a personalized model, which most typically consists of a non-uniform scaling (see, e.g., [38]) and a change of optimum fibre length.

Bolsterlee et al. [3] pointed out that many parameters in a model are interrelated. Adapting the model to the subject by scaling improves the anatomical resemblance between the model and the subject but may not improve force prediction. Unfortunately, it is not known how to adapt the other parameters. Several studies, e.g., [10,12,31], warn that attachment sites of muscles show high inter-subject variability, which may considerably affect muscle moment-arms because it has been shown that small differences in location of muscle attachment points often affect muscle force predictions to a great extent (see e.g., [4]).

Valente et al. [34] showed that representing a muscle, especially, a complex one, e.g., the gluteus medius, by a single line segment can produce errors up to 75% suggesting thus that the number of fibres in musculoskeletal models being in used might not be enough. Recently, Weinhandl & Bennett [37] confirmed that high number of fibres are required for the muscle surrounding the hip joint to provide an accurate estimation of joint contact forces. Modenese et al. [26] found out that representing the muscles surrounding the hip joint by fibres with none or a few via points only may limit the accuracy of hip contact force predictions.

To reduce the human effort associated with the construction of subject-specific musculoskeletal models, some researchers proposed algorithms to generate the fibres automatically providing that the surface model of a muscle is available [18,20,30]. The problem is how to update the shape of these fibres in reaction to the movement of bones. One approach to this problem is to express their vertices to be relative to the vertices of the surface mesh of the muscle, first, and then use some of the existing algorithms for surface mesh deformation proposed in the context of musculoskeletal modelling, e.g., [9,16,17,32].

In our conference paper [9], we proposed a new algorithm for muscle mesh deformation, based on position-based dynamics [28], and demonstrated its features on three hip muscles deforming during flexion of the right leg. In this paper, which is an extended version of that paper, we newly include:

- a description of the implementation details of our algorithm such as its initialization for muscle deformation, constraints calculations,
- a proposal of alternative algorithms for detecting the muscle points that should move with the bones,
- new experiments demonstrating the sensitivity of the results on its various parameters (e.g., anisotropy, number of iterations, resolution of the mesh),
- new experiments showing the lengths of fibres generated in the volume of hip muscles, and comparing them with those obtained by other approaches.

2 Position-Based Dynamics

Position-based dynamics (PBD), which is the core part of our approach, was firstly introduced in [28] as a fast, stable, and controllable alternative to mass-spring systems used in computer graphics algorithm. Since then, it has been further developed (e.g., [25] proposed recently some speed and accuracy improvements) and has found many (close to) real-time applications, not only in computer graphics, e.g., for simulations of cloth or fluids [33], but even in other domains. For example, Kotsalos et al. use PBD to model blood cells [24].

PBD represents a dynamic object, e.g., a muscle, by a set of N points, having associated mass and velocity, and a set of M constraints restricting the freedom of the movement of these points during the simulation. In their paper [28], Müller et al. presented the restraints to maintain distances among the points, the shape of the object and its volume, and to avoid collisions with other objects, however, one can use any constraint that is meaningful in their application context. Mathematically speaking, assuming that every point has the same mass,

the PBD method solves Eq. 1 that describes a movement of a single point $\mathbf{p}_i$ restricted by a constraint function C with cardinality n, where $\Delta\mathbf{p}_i$ denotes the difference in position of ith point and $\nabla_{p_i} C$ is the gradient of the function C with respect to point $\mathbf{p}_i$.

$$\Delta\mathbf{p}_i = -\frac{\nabla_{p_i} C\left(\mathbf{p}_1, \ldots, \mathbf{p}_n\right)}{\sum_{j=1}^{n} \left|\nabla_{p_j} C\left(\mathbf{p}_1, \ldots, \mathbf{p}_n\right)\right|^2} \cdot C\left(\mathbf{p}_1, \ldots, \mathbf{p}_n\right) \tag{1}$$

2.1 Distance Constraint

Distance constraint is restricting each model point to change the distance from the others in its neighbourhood. It is described by Eq. 2, where d is the original distance between points $\mathbf{p}_1$ and $\mathbf{p}_2$.

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

At this point, the gradient of this function has to be determined. Calculation procedure of determining the gradient of the vector norm is shown in (3).

$$\begin{aligned} \nabla_{\mathbf{p}_1} C\left(\mathbf{p}_1, \mathbf{p}_2\right) &= \nabla_{\mathbf{p}_1}\left(\left|\mathbf{p}_1 - \mathbf{p}_2\right| - d\right) \\ &= \frac{\left[\frac{\partial(p_{1x}-p_{2x})^2}{\partial p_{1x}} \;\; \frac{\partial(p_{1y}-p_{2y})^2}{\partial p_{1y}} \;\; \frac{\partial(p_{1z}-p_{2z})^2}{\partial p_{1z}}\right]}{2\left|\mathbf{p}_1 - \mathbf{p}_2\right|} \\ &= \frac{\mathbf{p}_1 - \mathbf{p}_2}{\left|\mathbf{p}_1 - \mathbf{p}_2\right|} = u \\ \nabla_{\mathbf{p}_2} C\left(\mathbf{p}_1, \mathbf{p}_2\right) &= \frac{\mathbf{p}_2 - \mathbf{p}_1}{\left|\mathbf{p}_1 - \mathbf{p}_2\right|} = -u \end{aligned} \tag{3}$$

Coincidentally, the result is the unit directional vector u of given edge.

2.2 Volume Constraint

Volume constraint restricts the object to change its volume during the simulation process. Assuming that this object is a triangular mesh model, the constraint function is:

$$C\left(\mathbf{p}_1, \ldots, \mathbf{p}_n\right) = \sum_{i=1}^{m} \left(\mathbf{p}_{t_1^i} \cdot \left(\mathbf{p}_{t_2^i} \times \mathbf{p}_{t_3^i}\right)\right) - V_0 \tag{4}$$

where m is the number of triangles forming the mesh, V_0 is its original volume, and $p_{t_j^i}$ is jth vertex of triangle i.

To obtain complete gradient of volume constraint function, all triangles are treated independently and their results are just summed together:

$$\nabla_{\mathbf{p}_i} C\left(\mathbf{p}_1, \ldots, \mathbf{p}_n\right) = \sum_{h=1}^{t} \mathbf{p}_j \times \mathbf{p}_k; \quad i \neq j \neq k \tag{5}$$

2.3 Local Shape Constraint

Above described constraints are not enough to prevent the triangular mesh model 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 model. This would, however, need to create a 3D mesh model first, which would be quite complex to do. Another option is to ensure that the local shape is maintained. To achieve this, the dihedral angles between neighbouring triangles should stay the same during deformation.

Equation 6 presents the local shape constraint function of triangles $\mathbf{p}_1, \mathbf{p}_2, \mathbf{p}_3$ and triangle $\mathbf{p}_2, \mathbf{p}_1, \mathbf{p}_4$ sharing points $\mathbf{p}_1$ and $\mathbf{p}_2$. In this equation, $\mathbf{n}_1$ and $\mathbf{n}_2$ are normal vectors of these triangles and φ_0 is the original dihedral angle between them. Gradients are defined in (7).

$$\begin{aligned} C\left(\mathbf{p}_1, \mathbf{p}_2, \mathbf{p}_3, \mathbf{p}_4\right) &= \arccos\left(\mathbf{n}_1 \cdot \mathbf{n}_2\right) - \varphi_0 \\ &= \arccos\left(\frac{\left(\mathbf{p}_2 - \mathbf{p}_1\right) \times \left(\mathbf{p}_3 - \mathbf{p}_1\right)}{\left|\left(\mathbf{p}_2 - \mathbf{p}_1\right) \times \left(\mathbf{p}_3 - \mathbf{p}_1\right)\right|_2}\right. \\ &\left. \cdot \frac{\left(\mathbf{p}_2 - \mathbf{p}_1\right) \times \left(\mathbf{p}_4 - \mathbf{p}_1\right)}{\left|\left(\mathbf{p}_2 - \mathbf{p}_1\right) \times \left(\mathbf{p}_4 - \mathbf{p}_1\right)\right|_2}\right) - \varphi_0 \end{aligned} \tag{6}$$

$$\begin{aligned} d &= \mathbf{n}_1 \cdot \mathbf{n}_2 \\ \nabla_{\mathbf{p'}_4} C &= -\frac{1}{\sqrt{1-d^2}}\left(\nabla_{\mathbf{p'}_4}\left(\mathbf{n}_2\right) \cdot \mathbf{n}_1\right) \\ \nabla_{\mathbf{p'}_3} C &= -\frac{1}{\sqrt{1-d^2}}\left(\nabla_{\mathbf{p'}_3}\left(\mathbf{n}_1\right) \cdot \mathbf{n}_2\right) \\ \nabla_{\mathbf{p'}_2} C &= -\frac{1}{\sqrt{1-d^2}}\left(\nabla_{\mathbf{p'}_2}\left(\mathbf{n}_1\right) \cdot \mathbf{n}_2 + \nabla_{\mathbf{p'}_2}\left(\mathbf{n}_2\right) \cdot \mathbf{n}_1\right) \\ \nabla_{\mathbf{p'}_1} C &= -\sum_{i=2}^{4} \nabla_{\mathbf{p'}_i} C \end{aligned} \tag{7}$$

3 Our Approach

The requested inputs of our approach are 1) a set of bones, each of which is represented by a triangular mesh and has an associated time-dependent transformation describing its movement, and 2) a muscle, also represented by a triangular mesh. We note that the first input is standard when creating any subject-specific musculoskeletal model. A muscle model is obtainable with a little effort from the medical images by segmentation (similarly as models of bones). Optionally, the user may specify a set of muscle fibres, represented by polylines, obtained, e.g., by Kohout & Kukačka [19], Kohout & Cholt [21], or Otake et al. [30] method. Furthermore, the user may also specify a set of attachment areas that describes the sites where the muscle attaches to the bones. As the muscle attachment sites

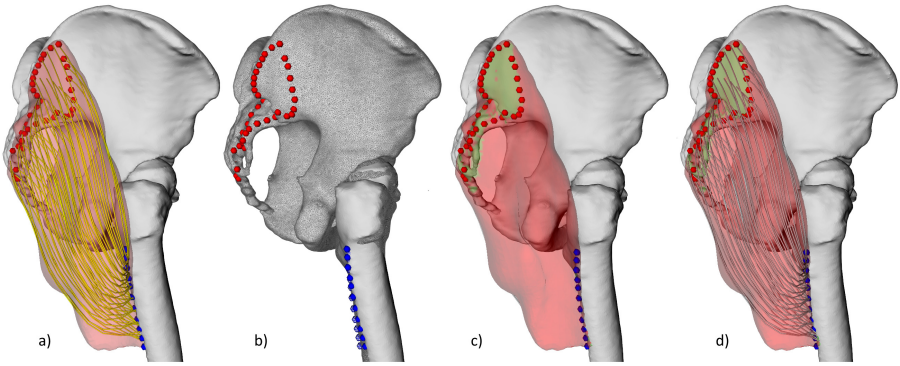

Fig. 2. Gluteus Maximus deformed by our approach: a) the input (origin and insertion attachment sites denoted by red and blue spheres), b) bones move from their initial rest-pose to the current pose (wireframe), c) the muscle surface adapted to the change of bones by PBD, d) the output. (color figure online)

are not apparent from the medical images, these are traditionally identified manually by an expert, typically as a set of landmarks fixed to the bones. Figure 2a shows an example of a typical input.

At each vertex of the muscle mesh, we create one PBD point with the mass being 1.0 and the initial velocity 0. For each pair of PBD points corresponding to the vertices connected in the muscle mesh by an edge, we establish the distance constraint (see Sect. 2.1) modified to support the anisotropic feature of muscles – see Sect. 3.1. Similarly, we create the local shape constraint (see Sect. 2.3) and the volume constraint (see Sect. 2.2). We note that we do not create any distance constraint between points of opposite sides of the muscle to avoid an unnatural change of the muscle shape during the simulation but rely solely on the latter two restraints in that.

We distinguish between two classes of PBD points: fixed and moveable, automatically detected as described in Sect. 3.3. A fixed point is bound to a single bone and moves with it at the beginning of the PBD simulation. The movement of the fixed points violates the equilibrium of the entire system, as described by the constraints, and the PBD attempts to restore it by updating, iteratively, the position of the moveable points while avoiding the penetration with the moved bones using the mechanism for collision detection and response described in Sect. 3.2 – see Fig. 2b,c.

Providing that the muscle fibres are specified, we compute the mean value coordinates of every vertex of polylines representing the fibres in the domain described by the triangular mesh of the muscle using the algorithm by Ju et al. [15]. Mathematically speaking, this operation maps the position of a muscle fibre vertex from E^3 to E^n, where n is the number of vertices of the muscle mesh. When the muscle surface deforms, the inverse mapping provides new positions of fibre vertices within the deformed domain (see Fig. 2d):

$$\mathbf{v'}_i = \sum_{j=1}^{n} w_j \cdot \mathbf{p'}_j \tag{8}$$

In the equation above, $\mathbf{v'}_i$ denotes the i-th fibre vertex, w_j are its mean value coordinates and $\mathbf{p'}_j$ is the position of the deformed muscle vertex $\mathbf{p}_j$.

The entire algorithm written in pseudocode is in Algorithm 1 and 2.

3.1 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 (see Eq. 2) with the result of the following equation:

Algorithm 1. Pre-processing stage of our algorithm.

procedure INIT(M, S_B, S_F, S_A) ▷ M is a muscle triangular mesh, S_F is a set of muscle fibres, S_B is a set of bones, and S_A is a set of attachment areas
  for all vertices $\mathbf{v}_i \in S_F$ **do**
    $\mathbf{w}_i = computeMVC(\mathbf{v}_i, M)$ ▷ compute the mean value coordinates
  end for

  for all bones $B \in S_B$ **do**
    $generateCollisionDataStructure(B)$ ▷ see Section 3.2
  end for

  for all vertices $\mathbf{p}_i \in M$ **do**
    $\mathbf{x}_i = \mathbf{p}_i$, $\mathbf{v}_i = 0$, $m_i = 1$ ▷ initialize a PBD point
  end for

  detect fixed points ▷ see Section 3.3

  for all edges $e_i \in M$ **do**
    $generateDistanceConstraint(e_i)$ ▷ compute the original distance d
    if $S_F = \emptyset$ **then**
      $k_i = 1$ ▷ no anisotropy used
    else
      $k_i = computeAnisotropyStiffness(e_i)$ ▷ see Section 3.1
    end if
    $generateLocalShapeConstraint(e_i)$ ▷ compute the dihedral angle φ_0
  end for

  $generateVolumeConstraint(M)$ ▷ compute the original volume V_0
end procedure

Algorithm 2. Runtime stage of our algorithm.

procedure EXECUTE($simFrame$) ▷ $simFrame$ is the index of simulation frame
  for all bones $B \in S_B$ **do** ▷ see also Algorithm 1
    $T = getTransform(B, simFrame)$ ▷ get the transformation matrix
    $transformMesh(B, T)$
  end for

  for all PBD points i **do**
    if isFixed(i) **then**
      $B = getAttachmentBoneForPoint(i)$
      $\mathbf{p}_i = transformPoint(\mathbf{x}_i, getTransform(B, simFrame))$
    else
      $\mathbf{v}_i = \mathbf{v}_i + \Delta t \cdot \mathbf{f}_{ext}(\mathbf{x}_i)/m_i$ ▷ update velocities by external forces
      $\mathbf{v}_i = \mathbf{v}_i \cdot c_{damp}$ ▷ apply some damping
      $\mathbf{p}_i = \mathbf{x}_i + \Delta t \cdot \mathbf{v}_i$
    end if
  end for

  loop solverIterations **times**
    for all edges $e_i \in M$ **do**
      $projectDistanceConstraintWithAnisotropy(e_i, k_i)$ ▷ updates $\mathbf{p}_i$
    end for
    $projectVolumeConstraint()$
    for all edges $e_i \in M$ **do**
      $projectLocalShapeConstraint(e_i)$
    end for
    for all vertices i **do**
      for all bones $B \in S_B$ **do**
        $T = getTransform(B, simFrame)$
        generateCollisionConstraints($B, T^{-1}, \mathbf{x}_i, \mathbf{p}_i$)
      end for
      $projectCollisionConstraints()$
    end for
  end loop

  for all verticies i **do**
    if $NotIsFixed(i)$ **then**
      $\mathbf{v}_i = \frac{\mathbf{p}_i - \mathbf{x}_i}{\Delta t}$ ▷ compute the velocity
      $\mathbf{x}_i = \mathbf{p}_i$ ▷ update the position
    end if
  end for

  for all vertices $\mathbf{p}_i \in M$ **do**
    $\mathbf{p}_i = \mathbf{x}_i$ ▷ update the muscle mesh
  end for
  for all vertices $\mathbf{v}_i \in S_F$ **do**
    $\mathbf{v}_i = reconstructPositionFromMVC(\mathbf{w}_i, M)$
  end for
end procedure

$$k_i = 1 - \mathbf{u}_i \cdot \mathbf{v}_i \tag{9}$$

The direction of ith 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_1 is equal to one and edge length will be preserved.

3.2 Collision Handling

The moving muscle and bones should not intersect each other. From several approaches to this issue we considered (see our conference paper [9]), we have opted for voxelization because of its simplicity and processing speed. In this approach, the bounding box of a bone is divided into a uniform grid of $n_x \times n_y \times n_z$ equally sized cells. For each triangle in the bone mesh, we detect the cells intersected by it and mark them as the boundary. Assuming that the mesh is closed, we mark the cells that are inside the bone using the flood-fill algorithm with 8-directions. All other cells are outside. Figure 3 shows the visualization of boundary cells when the constants n_x, n_y, and n_z are equal and when they are automatically determined from the sizes of the bounding box so that the overall number of cells is roughly equal to some given constant n_{max}.

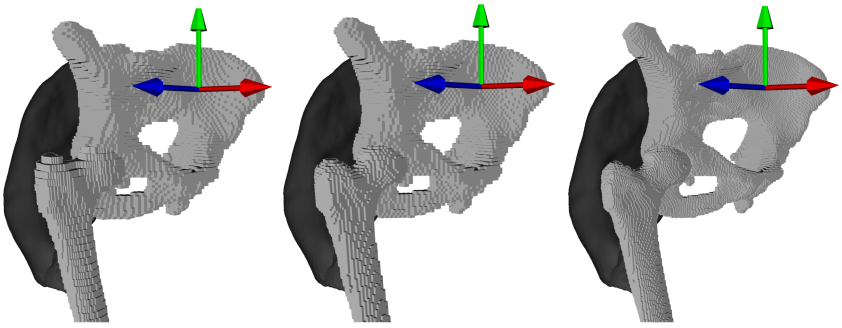

Fig. 3. Voxel representation of pelvis and femur. From left to right: $n_x = n_y = n_z = 64$ (262 144 cells, 256 KB min), $n_{max} = 262\ 144$ – pelvis $= 47 \times 64 \times 85$ (255 680 cells, 250 KB min) femur $= 42 \times 176 \times 34$ (251 328 cells, 245 KB min), $n_{max} = 8 \cdot 262\ 144 =$ 2 097 152 – pelvis $= 95 \times 128 \times 171$ (2 079 360 cells, ≈2 MB min) femur $= 85 \times 352 \times 69$ (2 064 480 cells, ≈2 MB min).

During the simulation, the algorithm identifies the cell in which a PBD point $\mathbf{p}_i$ lies. If this cell is outside the bone, the point does not collide with the bone. Otherwise, its position needs to be updated. Two scenarios have to be distinguished. In the first one, the muscle moves (e.g. because of surrounding forces) and hits a bone. As it is, the previous position of this point ($\mathbf{x}_i$) is outside the bone. The algorithm, therefore, traverses the collision data structure along the

line segment from $\mathbf{p}_i$ to $\mathbf{x}_i$ until it does not find an outside cell. If this cell is the cell of $\mathbf{x}_i$, $\mathbf{p}_i$ moves back to $\mathbf{x}_i$; otherwise, it moves to the point on the line segment where the traversal stopped.

In the second scenario, a bone moves into the muscle. Therefore, even the previous position of the point ($\mathbf{x}_i$) no longer lies outside the bone. We propose a solution where $\mathbf{p}_i$ moves to $\mathbf{x}_i$ transformed by the same transformation that caused the collision.

3.3 Detecting the Fixed Points

Assuming that the muscle is, in fact, a musculotendon unit, i.e., its surface touches the bones at the attachment sites, there are three approaches to detecting the muscle points that should be fixed to some bone and move with it, each of which has its pros and cons. In our previous work [9], we used the constructed data structure for collision detection also for the identification of the fixed points. However, recent analysis shows that this algorithm may inappropriately fix also the points that are close to some bone but should slide along it – see Fig. 4. That is the real reason for the unacceptable behaviour of the iliacus muscle during the flexion of the right leg reported in the original paper.

We, therefore, have experimented with another approach. We fix all points lying in the proximity of some bone, i.e., having their distances to the surface of a bone smaller or equal to a predefined threshold. An obvious choice is to compute the average length of edges in the muscle mesh and use it as this threshold. Figure 5, however, shows that the results are not very different from the results obtained by the original algorithm. Specification of the threshold value by the user may help. Nevertheless, this is very sensitive. For example,

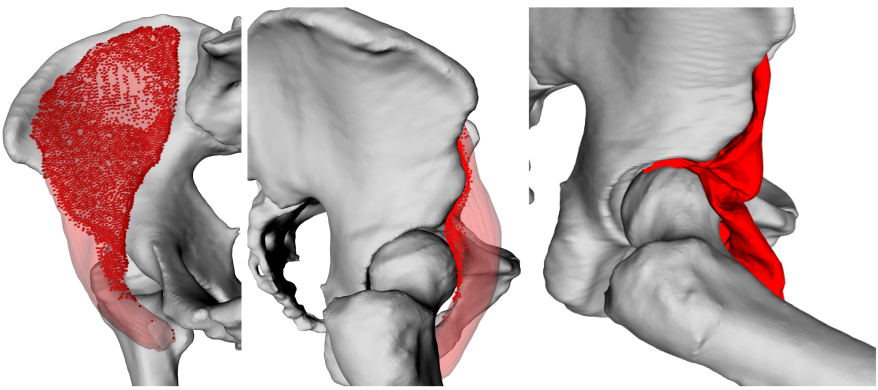

Fig. 4. Muscle vertices (red cubes) of Iliacus fixed to some bone, i.e., preserving their relative position to the bone during the simulation, as identified by the original CD-based algorithm ($n_x = n_y = n_z = 64$) causing an unsatisfactory result of the deformation (right). (Color figure online)

while 1 mm threshold seems perfect for gluteus maximus, for iliacus, a value less than 0.5 mm is needed to get something at least reasonable.

The third approach exploits the idea that muscle attachment areas are typically required for the construction of muscle fibres and, the user, therefore, have them readily available also for detection of the fixed points. We assume that an attachment area is defined by a set of landmarks that are fixed to a bone and, furthermore, they are specified in an order such that interconnecting every pair of adjacent landmarks by a line segment would produce a closed non-intersecting poly-line corresponding to the boundary of the attachment area. Following the idea described by Kohout & Kukačka in [19], we detect the patch on the muscle having the boundary corresponding to the boundary of the attachment area projected onto the muscle surface and fix all the points of this patch. As Fig. 5 demonstrates, this approach provides us with the best results.

4 Experimental Results

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.

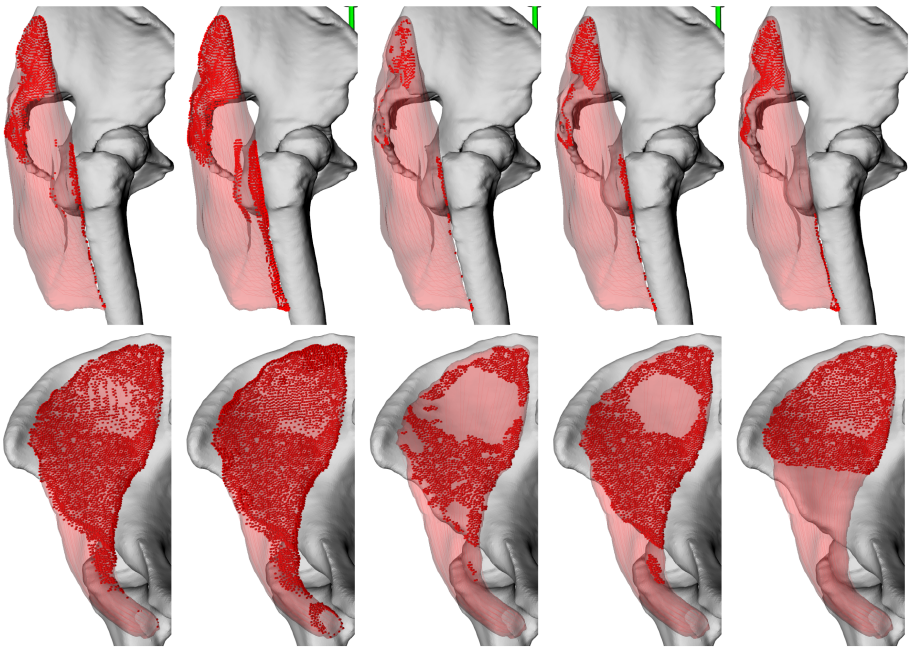

Fig. 5. Muscle vertices (red cubes) of gluteus maximus (top) and iliacus (bottom) fixed to some bone, i.e., preserving their relative position to the bone during the simulation, as identified by the original CD-based algorithm ($n_x = n_y = n_z = 64$), the muscle-bone proximity algorithm with the thresholds: average edge length, 0.5 mm, and 1 mm, and by the algorithm using muscle attachment areas input data. (Color figure online)

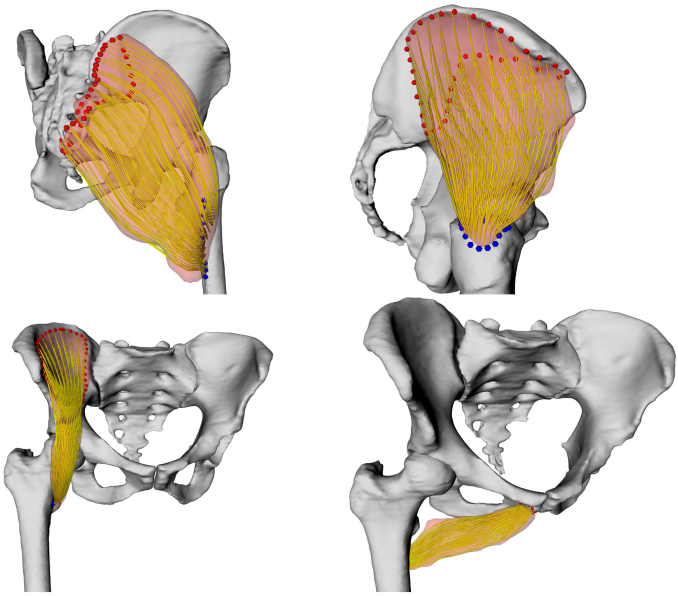

Fig. 6. Gluteus maximus (19 752 triangles (△)), gluteus medius (10 622 △), iliacus (13 858 △), and adductor brevis (17 124 △) decomposed into a set of 100 fibres composed of 15 line segments.

The complete data are publicly available in LHDL dataset [36] 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 [5]. The dataset also contains muscle attachment areas and geometrical paths of superficial fibres obtained from dissection [35].

To decompose the muscles into fibres, we use the Kohout & Kukačka method [19] with a slight modification: establishing the inter-contour correspondence is done by minimizing the sum of square distances between the corresponding points. This modification increases the robustness of the method even in cases when the user-specified number of straight-line segments per fibre is low.

We decomposed the surface meshes of gluteus maximus, gluteus medius, iliacus and adductor brevis into models of 100 fibres using a template with parallel fibres composed of 15 line segments – see Fig. 6. The decomposition took less than 50 ms in all cases on HP EliteDesk 800 G3 TWR (Intel Core i7-7700K @ 4.2 GHz, 64 GB RAM, Windows 10 64-bit).

Simulations of hip flexion (0° to 90°) were performed in steps of 2° 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.

The default parameters in all our experiments were: $n_{max} = 8 \cdot 64 \cdot 64 \cdot 64$, the damping constant $c_{damp} = 0.99$, anisotropy on, local shape constraint stiffness = 0.9 (i.e., the solver was allowed to violate this constraint to preserve the volume and avoid the penetration between the muscle and bones).

4.1 Number of Solver Iterations

In the first experiment, we investigated the influence of the number of iterations of constraint projections (see the loop on line 16 in Algorithm 2) on the quality of the results and overall time required for the simulation. From Fig. 7, it is apparent that the average displacement of points between individual iterations quickly decreases. In a few iterations, it drops below 0.1 mm; with just 10 iterations it is below 0.01 mm.

Average time required by one simulation step (Algorithm 2) on HP EliteDesk 800 G3 TWR (Intel Core i7-7700K @ 4.2 GHz, 64 GB RAM, Windows 10 64-bit) using our, mostly unoptimized, C++ implementation is in Table 1.

Table 1. Times needed for one simulation step on average for adductor brevis using various number of PBD solver iterations (NoIters).

NoIters	1	3	5	10	25	50	100
Time [ms]	53.04	62.55	74.66	104.96	193.66	441.72	658.71

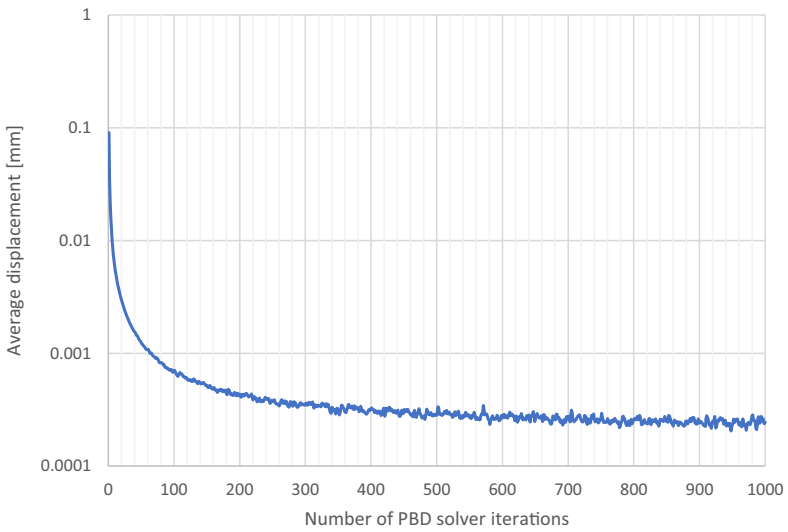

Fig. 7. The average displacement of points of adductor brevis between individual iterations of the PBD solver during the hip flexion. Note the logarithmic scale on the y-axis.

Figure 8 brings a visual comparison of the results obtained using a various number of iterations. An unrealistic bending of the muscle is apparent, especially when using a few iterations only. This behaviour has three reasons. First, the speed of the femur bone is quite high; it is 2° per simulation step, which represents the movement of the fixed points of 3.78–6.75 mm. Next, at the beginning of the simulation, the moving femur hits an unfixed part of the muscle giving it a large velocity pulling it in the direction opposite to the natural movement. Finally, the muscle mesh contains 17126 triangles, i.e., it is pretty accurate, and, therefore, a lot of iterations are required to propagate the movement from the points on the femur to those on the pelvis.

Hence, we reduced the number of triangles using the quadric edge collapse decimation implemented in MeshLab software down to 3000 (L1) and 1000 (L2). Not only visual realism improves, as Fig. 9 illustrates, but also the overall required time decreases since there are less PBD points and consequently also fewer constraints to satisfy. For L2 mesh, 1000 iterations need 349.80 ms per simulation step on average, which is even faster than 100 iterations for the original, high-resolution mesh. Naturally, this higher number of iterations improves visual appearance considerably. We note, however, that increasing the number of iterations further, e.g., to 10000, does not bring any substantial change.

4.2 Fixed Points

Figure 8 also demonstrates the effect of the algorithm used to detect the points to fix on the results of the deformation. Due to inaccuracies during the extraction of the musculotendon unit, only a very small part of the adductor brevis muscle is touching the femur. When using the original algorithm, which exploits the collision detection mechanism, a significant area on the muscle is, therefore, not fixed. As a result, the deformation algorithm produces the mesh with an unrealistic sharp spike. There is no such issue with the detection algorithm exploiting the knowledge of muscle attachment areas.

A different case happens with the iliacus muscle – see Fig. 10. Despite the relatively high resolution of the voxel data structure, many muscle points in proximity of the femur ball are fixed incorrectly to the femur. As a result, this part of the muscle moves unrealistically into the narrow space between the femur and pelvis. Using the attachment areas improves the situation but only slightly because the points in the proximity of the femur ball typically collide with the coarse voxel representation of the femur and they are, therefore, transformed using the same transformation. After turning this collision handling mechanism off, the algorithm provides us with acceptable results with a small muscle-bone penetration.

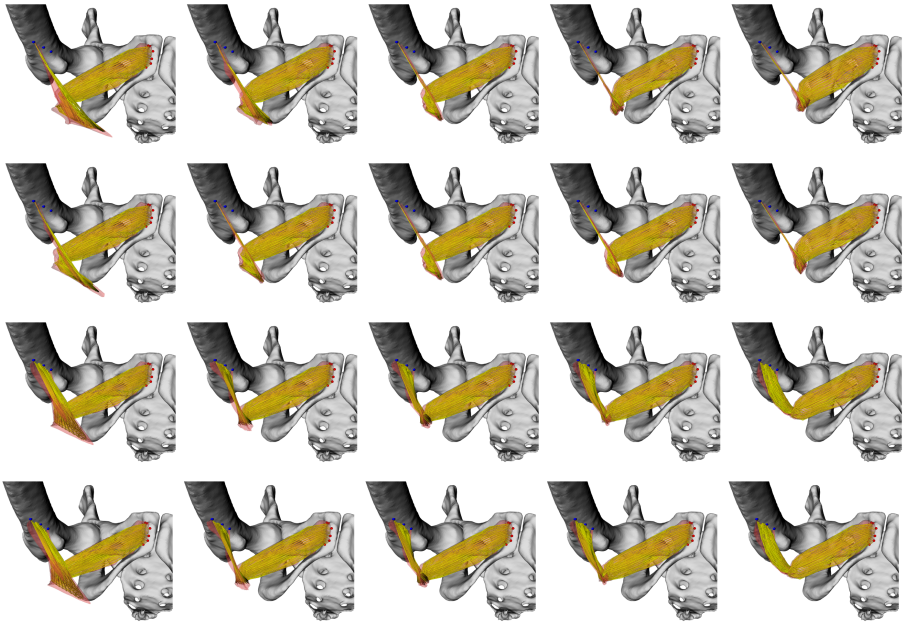

Fig. 8. Adductor brevis at flexion of 40° using 1, 3, 5, 10 or 100 PBD solver iterations (from left to right) with anisotropy off (odd rows) and on (even rows) when fixing the points by the original CD-based algorithm with $n_{max} = 8 \cdot 262\,144$ (the first two rows) and by the algorithm using muscle attachment areas (the last two rows).

4.3 Anisotropy, Volume and Other Constraints

The impact of the anisotropy on the results is apparent in Fig. 8. Surprisingly, it is barely observable. Most probably, this is because the other constraints (especially the volume constraint) play a dominant role. Volume preservation constraint was tested by determining ratio between both original and actual volumes. Figure 11 show us the volume preservation results. As we can see, the volume is well preserved (the error is less than 1% in all cases). Other quantitative tests, e.g., preservation of the dihedral angles between two adjacent triangles and average edge extension, are presented in our original conference paper [9].

4.4 Muscle Fibre Lengths

Last but not least, we analyzed the lengths of fibres reconstructed at the end of the deformation step. To remove any noise that might be present in the data, we performed a smoothing process, repeated five times, that updates the length l_i according to the equation: $l'_i = (l_{i-1} + 4 \cdot l_i + l_{i+1})/6$. The results are present in Figs. 12, 13, and 14. Both gluteus maximus and gluteus medius behave during the hip flexion as expected. The lengths of all the gluteus maximus fibres increase. In the case of the gluteus medius, only the surface fibres extend, while the deep

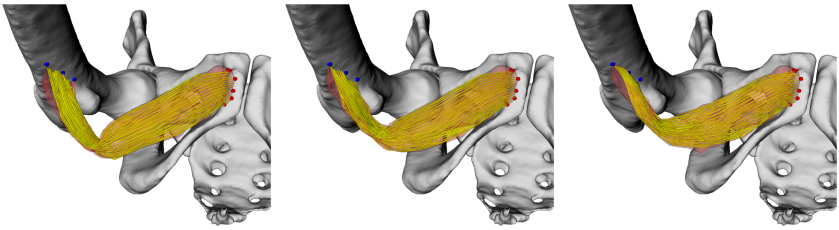

Fig. 9. Adductor brevis at flexion of 40° using 100 PBD solver iterations, anisotropy on, fixing the points at muscle attachment areas when a surface mesh with 17 124, 3 000, and 1 000 triangles is used.

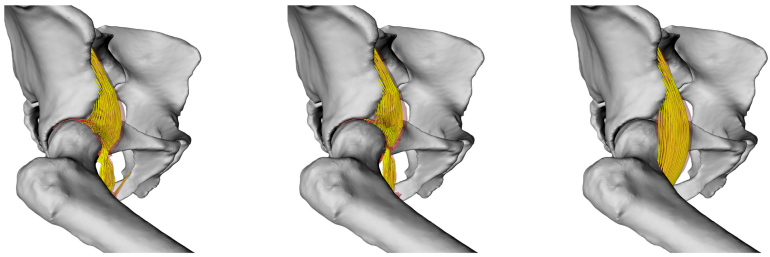

Fig. 10. Iliacus at flexion of 40° using 5 PBD solver iterations, anisotropy on, fixing the points by the original CD-based algorithm with $n_{max} = 8 \cdot 262\,144$ (left) and by the algorithm using muscle attachment areas with (middle) and without (right) collision handling when a bone hits the muscle.

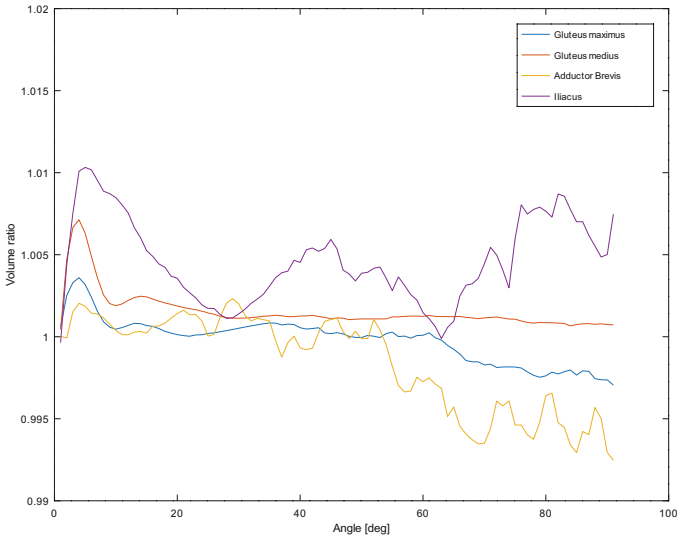

Fig. 11. Volume preservation during hip flexion using 3 PBD solver iterations.

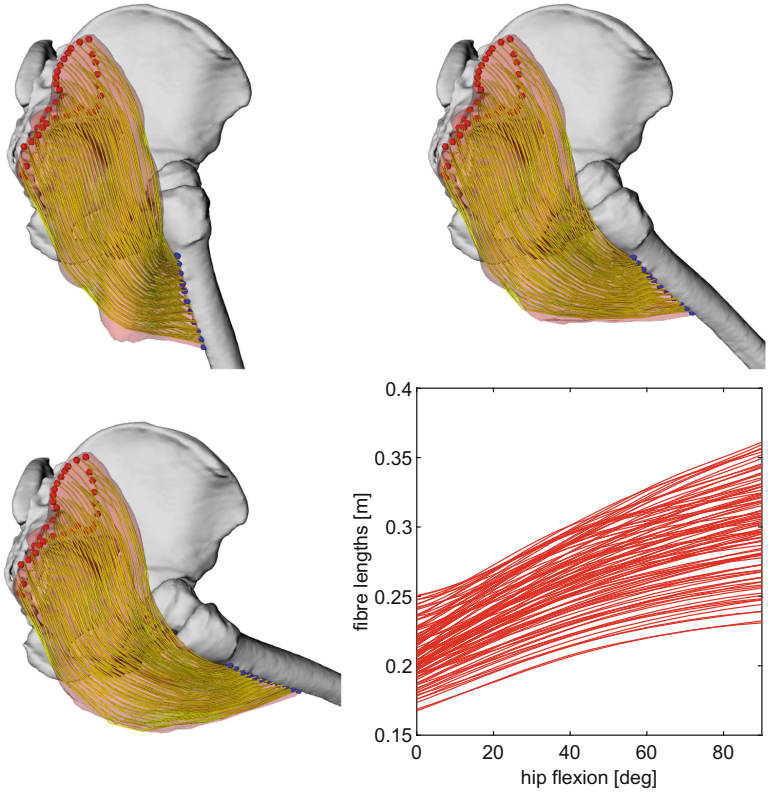

Fig. 12. Total length of each individual fibre during simulation in the gluteus maximus muscle. The visual results at 20°, 50°, and 70° are shown for illustration.

fibres contract. For the iliacus muscle, we can observe an unrealistic increase in the lengths when the flexion is greater than 70°, which is caused by the above-described issue of pushing a part of the muscle into the joint space.

4.5 Deformation 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.

Using the collision detection structure with $n_x = n_y = n_z = 64$ and three PBD solver iterations, we measured the processing speed of our implementation. All tests were performed on Intel® Core™ i7-4930K 3.40 GHz CPU, Radeon HD 8740 GPU and WDC WD40EURX-64WRWY0 4TB HDD. The results, given in FPS (Frames Per Second), i.e., the number of simulation steps per second, are listed in Table 2. As it can be seen, the FPS strictly depends on number of triangles (Spearman's $\rho = -1$). The more triangles is used, the slower the method

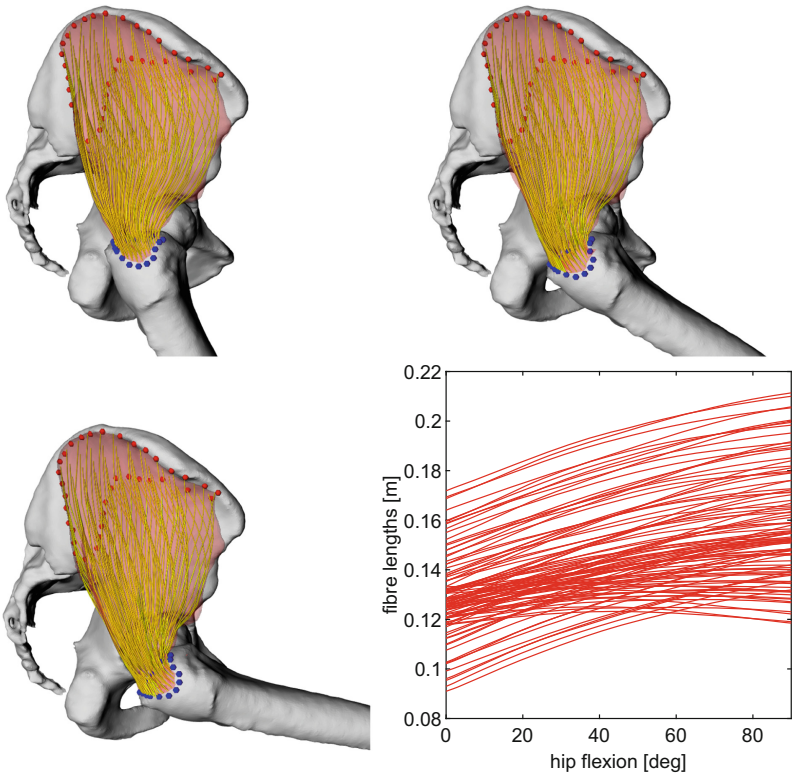

Fig. 13. Total length of each individual fibre during simulation in the gluteus medius muscle. The visual results at 20°, 50°, and 70° are shown for illustration.

Table 2. 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

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

5 Discussion

In the past, several algorithms for the deformation of the surface mesh of a muscle were proposed. Most of these algorithms, however, have unreal requirements on the input, e.g., [16,17] rely on existence of a muscle skeleton (centroid)

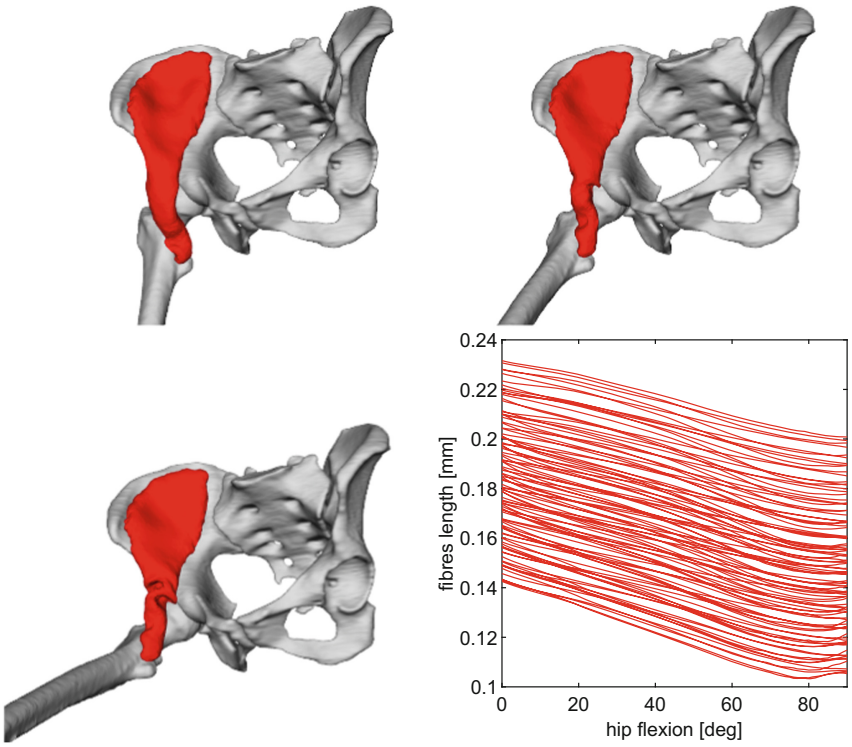

Fig. 14. Total length of each individual fibre during simulation in the iliacus muscle. The visual result at 20°, 50°, and 70° are shown for illustration. For clarity, we do not show the fibres. Readers are referred to Fig. 10 to see the produced fibres of the iliacus muscle.

having known a physiologically correct deformation, or they ignore important physiological properties such as impenetrability with bones and other muscles (e.g., [17,32]), muscle volume preservation, and anisotropy of muscles during their contraction.

Romeo et al. [32] independently to our work developed an approach similar to ours. The main differences are as follows. First, the authors build a complex internal muscle structure to better preserve the shape and volume of the muscle, while we work with the surface geometry only. Next, they do not include any mechanism to prevent penetration of muscles and bones, relying thus on manually defined various mesh-to-mesh constraints, which not only complicates the setup but also does not guarantee impenetrability. We implemented a simple and fast collision handling that avoids muscle-bone penetration in most cases. Finally, their aim is to have a visually plausible skin deformation but what is going on inside the body is not of their interest. We, on the other hand, focus on the representation of muscles for mechanical assessments.

Janak et al. [14] proposed a technique based on the mass-spring system to deform the fibres while preventing their penetration with bones and fibres of other muscles. To get reasonable results, a lot of particles are required, which causes high time and memory complexity. More importantly, the muscle volume is not preserved. This could be probably solved using the approach described in [13], 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 approach, 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.

The most complex way to solve muscle dynamics is by using the finite element method (FEM). This approach is physically the most accurate one if the muscle is well discretized (see e.g., [7]). 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 VFX[1] plugin for Maya, and in muscle physiology research, see e.g. [29] or [23]. In comparison with these methods, our method is quite simple to set up and runs fast providing the promising results in most cases.

Recently, Modenese & Kohout [27] presented a simple method that calculates the kinematic position of a vertex of the fibre as a linear combination of the transformations of its rest-pose position with respect to the bones with the attachment sites of the muscle this fibre belongs to, whereas the blending weight is chosen as a function of the relative distance of this vertex from the origin point of the fibre with one user-specific parameter to minimize the penetration of the fibre with bones. Using the approach described in [18] to highly discretize the muscles of pelvic region (up to 100 fibres of 15 line segments), the fibres' moment arms of hip flexion, adduction, and internal rotation were validated against measurements and models of the same muscles from the literature with promising outcomes. Nevertheless, extending the method for muscles wrapping around multiple bones, such as *rectus femoris*, is not straightforward. Furthermore, a muscle-bone penetration cannot be avoided and in the case of the iliacus muscle, the fibres are also unrealistically pushed into the hip joint. Similarly to [14], the volume of a muscle cannot be preserved.

We compared the length of the fibres produced by Modenese & Kohout [27] with those produced by our approach using the same data. Figure 15 shows a good match between the results for the gluteus medius and the iliacus. A significant difference is apparent for the gluteus maximus. The range of lengths of our fibres is much bigger than theirs, whereas our fibres tend to be longer. One of

[1] https://zivadynamics.com/.

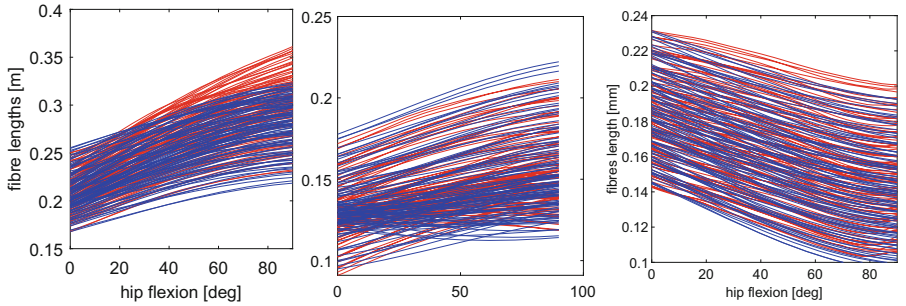

Fig. 15. Comparison of the lengths of the fibres of the gluteus maximus (left), gluteus medius (middle), and iliacus (right) muscle produced by our approach (red) and by the approach described in [27] (blue). (Color figure online)

the reasons for this difference is that our approach guarantees impenetrability between muscle and bones. As a result, all the fibres have to wrap around the joint and, naturally, they must be longer than the fibres produced by the other approach, where some fibres penetrate the femur in extreme positions. The volume preservation constraint prevents the flattening of the muscle at the greater trochanter of the femur, which means that the surface fibres are more distant from the bone than in the other approach. Consequently, they are longer.

There are some limitations of the proposed approach. First of all, the experiments have shown that detecting the muscle points that should move with bones exploiting the information about attachment areas of the muscle is superior in most cases when compared with proximity or collision-based detection. The muscle attachment sites, however, cannot be extracted automatically from the medical images and their manual specification, by an expert in anatomy, is time-consuming. Nevertheless, Fukuda et al. [10] proposed an approach to the automatic estimation of the muscle attachments that is based on applying a non-rigid transformation of the surface model of a normalized (average) bone with a normalized attachment site specified onto the surface model of the subject-specific bone. When the normalized attachment site is obtained from a probabilistic atlas built as suggested by the authors, the estimations are quite accurate, with dice coefficients reaching up to 70%.

Next, the proposed collision handling is inaccurate, which leads to an appearance of sharp spikes on the surface of the muscle, especially, when using a coarse voxel representation of bones. Naturally, as the memory complexity of this representation grows cubically, it is obvious that using a refined voxel representation is impractical. In the scenario when a bone moves into a muscle, setting the velocities of the colliding points to zero instead of using the formula on line 34 (in Algorithm 2) could help.

Finally, the results are very sensitive to the settings of the parameters. Fortunately, as the simulation runs in real-time, even using an unoptimized sequential

implementation, the user may tune the values of these parameters until they are satisfied with the visual output of our approach.

6 Conclusion and Future Work

The presented approach is capable of performing a visually plausible and physically correct real-time deformation of muscles represented by triangular meshes in most cases we tested. The main issue is with the iliacus muscle, which (when deformed) looks unrealistic. Nevertheless, the qualitative and quantitative results (e.g., the length of the fibres produced in the volume of the deformed muscle) are comparable with the other state-of-the-art methods. In the future, the iliacus muscle deformation will be further analyzed and the issue with muscle tissue entering the joint is to be solved.

The implementation is written in C++ and partially included in OpenSim (a state-of-the-art simulation software) as a plugin. Its source code is available at https://github.com/cervenkam/muscle-deformation-PBD.

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

References

1. Arnold, E.M., Ward, S.R., Lieber, R.L., Delp, S.L.: A model of the lower limb for analysis of human movement. Ann. Biomed. Eng. **38**(2), 269–279 (2009). https://doi.org/10.1007/s10439-009-9852-5. http://www.ncbi.nlm.nih.gov/pmc/articles/PMC2903973/
2. Audenaert, A., Audenaert, E.: Global optimization method for combined spherical-cylindrical wrapping in musculoskeletal upper limb modelling. Comput. Methods Programs Biomed. **92**(1), 8–19 (2008). https://doi.org/10.1016/j.cmpb.2008.05.005. http://www.ncbi.nlm.nih.gov/pubmed/18606476
3. Bolsterlee, B., Veeger, D.H.E.J., Chadwick, E.K.: Clinical applications of musculoskeletal modelling for the shoulder and upper limb. Med. Biol. Eng. Comput. **51**(9), 953–963 (2013). https://doi.org/10.1007/s11517-013-1099-5
4. Carbone, V., van der Krogt, M., Koopman, H., Verdonschot, N.: Sensitivity of subject-specific models to errors in musculo-skeletal geometry. J. Biomech. **45**(14), 2476–2480 (2012). https://doi.org/10.1016/j.jbiomech.2012.06.026
5. Cignoni, P., Callieri, M., Corsini, M., Dellepiane, M., Ganovelli, F., Ranzuglia, G.: MeshLab: an open-source mesh processing tool. Computing **1**, 129–136 (2008). https://doi.org/10.2312/LocalChapterEvents/ItalChap/ItalianChapConf2008/129-136
6. Delp, S.L., Loan, J.P., Hoy, M.G., Zajac, F.E., Topp, E.L., Rosen, J.M.: An interactive graphics-based model of the lower extremity to study orthopaedic surgical procedures. IEEE Trans. Biomed. Eng. **37**(8), 757–767 (1990). https://doi.org/10.1109/10.102791
7. Delp, S.: Three-dimensional representation of complex muscle architectures and geometries 1. Ann. Biomed. Eng. **33**, 1134 (2005). https://doi.org/10.1007/s10439-005-1433-7

8. Delp, S.L., Loan, J.P.: A computational framework for simulating and analyzing human and animal movement. Comput. Sci. Eng. **2**(5), 46–55 (2000)
9. Červenka, M., Kohout, J.: Fast and realistic approach to virtual muscle deformation. In: Proceedings of the 13th International Joint Conference on Biomedical Engineering Systems and Technologies. SCITEPRESS - Science and Technology Publications (2020). https://doi.org/10.5220/0009129302170227
10. Fukuda, N., et al.: Estimation of attachment regions of hip muscles in CT image using muscle attachment probabilistic atlas constructed from measurements in eight cadavers. Int. J. Comput. Assist. Radiol. Surg. **12**(5), 733–742 (2017). https://doi.org/10.1007/s11548-016-1519-8
11. Garner, B., Pandy, M.: The obstacle-set method for representing muscle paths in musculoskeletal models. Comput. Methods Biomech. Biomed. Eng. **3**(1), 1–30 (2000)
12. Herteleer, M., et al.: Variation of the clavicle's muscle insertion footprints - a cadaveric study. Sci. Rep. **9**(1), 1–8 (2019). https://doi.org/10.1038/s41598-019-52845-8
13. Hong, M., Jung, S., Choi, M.H., Welch, S.: Fast volume preservation for a mass-spring system. IEEE Comput. Graph. Appl. **26**, 83–91 (2006). https://doi.org/10.1109/MCG.2006.104
14. Janák, T., Kohout, J.: Deformable muscle models for motion simulation. In: Proceedings of the 9th International Conference on Computer Graphics Theory and Applications. pp. 301–311. SCITEPRESS - Science and and Technology Publications (2014). https://doi.org/10.5220/0004678903010311
15. Ju, T., Schaefer, S., Warren, J.: Mean value coordinates for closed triangular meshes. ACM Trans. Graph. **24**(3), 561–566 (2005). http://portal.acm.org/citation.cfm?doid=1073204.1073229
16. Kellnhofer, P., Kohout, J.: Time-convenient deformation of musculoskeletal system. In: ALGORITMY 2012, 19th Conference on Scientific Computing, Vysoke Tatry, Slovakia, 09–14 Sep 2012, pp. 239–249. Slovak Univ Technology, Bratislava (2012)
17. Kohout, J., et al.: Patient-specific fibre-based models of muscle wrapping. Interface Focus **3**(2), 20120062 (2013). https://doi.org/10.1098/rsfs.2012.0062
18. Kohout, J., Kukačka, M.: Real-time modelling of fibrous muscle. Comput. Graph. Forum **33**(8), 1–15 (2014). https://doi.org/10.1111/cgf.12354
19. Kohout, J., Kukačka, M.: Real-time modelling of fibrous muscle. In: Computer Graphics Forum [18], pp. 1–15. https://doi.org/10.1111/cgf.12354
20. Kohout, J., Cholt, D.: Automatic reconstruction of the muscle architecture from the superficial layer fibres data. Comput. Methods Programs Biomed. **150**, 85–95 (2017). https://doi.org/10.1016/j.cmpb.2017.08.002
21. Kohout, J., Cholt, D.: Automatic reconstruction of the muscle architecture from the superficial layer fibres data. In: Computer Methods and Programs in Biomedicine [20], pp. 85–95. https://doi.org/10.1016/j.cmpb.2017.08.002
22. Kohout, J., Clapworthy, G.J., Martelli, S., Viceconti, M.: Fast realistic modelling of muscle fibres. In: Csurka, G., Kraus, M., Laramee, R.S., Richard, P., Braz, J. (eds.) VISIGRAPP 2012. CCIS, vol. 359, pp. 33–47. Springer, Heidelberg (2013). https://doi.org/10.1007/978-3-642-38241-3_3
23. Kojic, M., Mijailovic, S., Zdravkovic, N.: Modelling of muscle behaviour by the finite element method using Hill's three-element model. Int. J. Numer. Meth. Eng. **43**(5), 941–953 (1998). https://doi.org/10.1002/(SICI)1097-0207(19981115)43:5⟨941::AID-NME435⟩3.0.CO;2-3

24. Kotsalos, C., Latt, J., Chopard, B.: Bridging the computational gap between mesoscopic and continuum modeling of red blood cells for fully resolved blood flow. J. Comput. Phys. **398**, 108905 (2019). https://doi.org/10.1016/j.jcp.2019.108905. cited By 0
25. Macklin, M., et al.: Small steps in physics simulation. In: Proceedings of the 18th Annual ACM SIGGRAPH/Eurographics Symposium on Computer Animation, SCA 2019, pp. 2:1–2:7. ACM, New York (2019). https://doi.org/10.1145/3309486.3340247
26. Modenese, L., Gopalakrishnan, A., Phillips, A.: Application of a falsification strategy to a musculoskeletal model of the lower limb and accuracy of the predicted hip contact force vector. J. Biomech. **46**(6), 1193–1200 (2013). https://doi.org/10.1016/j.jbiomech.2012.11.045
27. Modenese, L., Kohout, J.: Automated generation of three-dimensional complex muscle geometries for use in personalised musculoskeletal models. Ann. Biomed. Eng. **48**, 1793–1804 (2020). https://doi.org/10.1007/s10439-020-02490-4
28. Müller, M., Heidelberger, B., Hennix, M., Ratcliff, J.: Position based dynamics. J. Vis. Commun. Image Represent. **18**, 109–118 (2007). https://doi.org/10.1016/j.jvcir.2007.01.005
29. Oberhofer, K., Mithraratne, K., Stott, N.S., Anderson, I.A.: Anatomically-based musculoskeletal modeling: prediction and validation of muscle deformation during walking. Vis. Comput. **25**(9), 843–851 (2009). https://doi.org/10.1007/s00371-009-0314-8
30. Otake, Y., et al.: Patient-specific skeletal muscle fiber modeling from structure tensor field of clinical CT images. In: Descoteaux, M., Maier-Hein, L., Franz, A., Jannin, P., Collins, D.L., Duchesne, S. (eds.) MICCAI 2017. LNCS, vol. 10433, pp. 656–663. Springer, Cham (2017). https://doi.org/10.1007/978-3-319-66182-7_75
31. Pellikaan, P., et al.: Evaluation of a morphing based method to estimate muscle attachment sites of the lower extremity. J. Biomech. **47**(5), 1144–1150 (2014). https://doi.org/10.1016/j.jbiomech.2013.12.010
32. Romeo, M., Monteagudo, C., Sánchez-Quirós, D.: Muscle and fascia simulation with extended position based dynamics. Comput. Graph. Forum **39**(1), 134–146 (2019). https://doi.org/10.1111/cgf.13734
33. Shao, X., Liao, E., Zhang, F.: Improving SPH fluid simulation using position based dynamics. IEEE Access **5**, 13901–13908 (2017). https://doi.org/10.1109/ACCESS.2017.2729601
34. Valente, G., Martelli, S., Taddei, F., Farinella, G., Viceconti, M.: Muscle discretization affects the loading transferred to bones in lower-limb musculoskeletal models. Proc. Inst. Mech. Eng. Part H J. Eng. Med. **226**(2), 161–169 (2012)
35. Van Sint Jan, S.: Introducing anatomical and physiological accuracy in computerized anthropometry for increasing the clinical usefulness of modeling systems. Crit. Rev. Phys. Rehabil. Med. **17**, 149–174 (2005). https://doi.org/10.1615/CritRevPhysRehabilMed.v17.i4.10
36. Viceconti, M., Clapworthy, G., Van Sint Jan, S.: The virtual physiological human - a European initiative for in silico human modelling. J. Physiol. Sci.: JPS **58**, 441–446 (2008). https://doi.org/10.2170/physiolsci.RP009908
37. Weinhandl, J.T., Bennett, H.J.: Musculoskeletal model choice influences hip joint load estimations during gait. J. Biomech. **91**, 124–132 (2019). https://doi.org/10.1016/j.jbiomech.2019.05.015
38. Zhao, Y., et al.: Laplacian musculoskeletal deformation for patient-specific simulation and visualisation. In: 2013 17th International Conference on Information Visualisation. IEEE (2013). https://doi.org/10.1109/iv.2013.67