Proceedings of ALGORITMY 2012
pp. 239–249

TIME-CONVENIENT DEFORMATION OF MUSCULOSKELETAL SYSTEM*

PETR KELLNHOFER† AND JOSEF KOHOUT‡

Abstract. The musculoskeletal modelling and simulation is an essential step in the process of looking for an optimal strategy to provide patients suffering from various musculoskeletal disorders, such as osteoporosis, with better health care. In our previous work, we proposed a deformation method suitable for clinical practise that deforms each muscle represented by a surface mesh according to assigned skeleton action lines and preserves its inner volume at the same time. It is built on combination of linear constraints for surface description together with relation of surface to control skeleton and non-linear constraint of volume preservation. It uses Gauss-Newton based iterative solver to find energy minimum fulfilling these conditions. It gains extra performance from exploiting the coarse outer hull for potentially slow and numerically unstable calculations. It achieves excellent ratios of volume preservation and maintains reasonable times in hundreds of milliseconds for our typical meshes, but since each mesh is deformed independently to others, it is unable to provide deformation of multiple interacting meshes without danger of their mutual intersections.

This is why a new modification of the method is introduced in this paper that alters both constraint formulation and the iterative solver algorithm to fix and prevent major intersections between mesh surfaces. It detects and prevents initial intersections in the starting pose of meshes and then prevents new intersections during the solution by constraining the modification step of meshes. It however still considers importance of volume preservation and tries to minimize effect of changes on its maintenance. The method was implemented in C++ language and VTK framework and integrated to our human body framework. The results of application to medical data we use show that despite of a few open issues, the proposed technique has its merit.

Key words. skeleton driven deformation, Laplacian coordinates, volume preservation, muscle modelling, intersection prevention, VTK framework

AMS subject classifications. 68U05, 65D18

1. Introduction.

In this paper, we present a modification of the mesh deformation algorithm described in our previous paper [4]. It incorporates intersection prevention to allow for processing of multiple interacting inputs simultaneously while still maintaining features of the original method. Those are deformation steered by external skeleton lines, maintaining the inner volume of individual models and achieving interactive execution times.

As far as we know there is no similar approach solving our problem and therefore we only provide comparison to the original version of the method. See the previous paper [4] for comparison of original method and its alternatives.

The goal of our research is application of the algorithm on data of the human body framework developed as a part of the European Union research project *VPHOP: Osteoporotic virtual physiological human* (*FP7-ICT-223865*) where it will allow simulation of bones and meshes in the moving human body for medical purposes.

*This work was supported by the Information Society Technologies Programme of the European Commission under the project VPHOP (FP7 ICT-223865) and by the Ministry of Education of The Czech Republic under the project 7E11016.

†Department of Computer Science and Engineering, University of West Bohemia, Plzeň, Czech Republic keni@students.zcu.cz.

‡Department of Computer Science and Engineering, University of West Bohemia, Plzeň, Czech Republic besoft@kiv.zcu.cz

2. Deformation of single model.

2.1. Introduction. The original method described in [4] performed a skeleton driven deformation with volume preservation on a single mesh model. It is based on method described in [2].

The deformation is governed by input triangular mesh and initial and final skeleton polylines with their mutual relations defined by control point order. There may be one or more skeletons for the mesh and they may lie both inside and outside of the object. At the beginning, low polygon coarse mesh is built from the input mesh so that it forms outer hull. Then mean value coordinates in the coarse mesh are found for each vertex of the original mesh using the algorithm described in [3]. This provides mapping from coarse mesh back to original resolution. Thanks to this, the deformation can be calculated on the coarse mesh thus saving both memory and computing power. Only final steps will be performed on the detail mesh to maintain high precision of the volume preservation.

2.2. Constraints. Three constraints are expressed using mathematical equations. The first constraint is linear and expresses the shape of the mesh using Laplacian coordinates. The second constraint is also linear and expresses directions of the deformation. The last constraint is non-linear and is used to maintain the original volume of the mesh according to the equation 2.6.

For each vertex $\overrightarrow{x_i}$ of the coarse mesh, the Laplacian condition expresses shape of local neighbourhood by distributing its coordinates relatively to linear combination of its neighbours $N(\overrightarrow{x_i})$:

$$\delta(\overrightarrow{x_i}) = \sum_{j=0}^{|N(\overrightarrow{x_i})|} w_j \cdot \overrightarrow{x_j} \tag{2.1}$$

where w_j is a weight of a respective neighbour.

We use cotangent weights where the weight is given by cotangent of angles at vertices opposing the edge $(\overrightarrow{x_i}, \overrightarrow{x_j})$:

$$w_i = \cot|\angle \overrightarrow{x_i}\overrightarrow{x_{i_{t_1}}}\overrightarrow{x_{i_s}}| + \cot|\angle \overrightarrow{x_i}\overrightarrow{x_{i_{t_2}}}\overrightarrow{x_{i_s}}| \tag{2.2}$$

When written for each vertex individually, the set of equations is built. This can be written in matrix form as

$$\mathcal{L}X = \delta(X) \tag{2.3}$$

As the right-hand side of this equation consists of directions, the relation is not invariant to rotation. Hence, all directions must be pre-rotated. Nearest rest pose skeleton segment for each vertex is found and the rotation for the operator is specified by the change of the direction from the located segment and its image into the target pose skeleton.

The second constraint is defined to preserve the mutual relationship between the skeleton and the mesh. Each original and final pose skeleton polylines are sampled, so pairs of points representing the skeleton movement are gained. Then each original pose point $\overrightarrow{s_j}$ is tied with the coarse mesh vertices $\overrightarrow{x_j}$ using mean value coordinates

k_j [3] by applying the same approach as for the original and coarse mesh relation. This results in

$$\overrightarrow{s_i} = \sum_{j=1}^{N} k_j \cdot \overrightarrow{x_j} \tag{2.4}$$

When the absolute coordinates of an original skeleton point $\overrightarrow{s_i}$ are replaced by the coordinates of the matching final skeleton point $\overrightarrow{s_i}'$, this equation becomes invalid in the initial state of deformation where $\overrightarrow{x_j} \equiv \overrightarrow{x_j}^0$ so the solver will be forced to find $\overrightarrow{x_j}^k$ fulfilling the constraint. This will enforce the deformation. Expressing the relation for each vertex gives a set of linear equations:

$$SX = s' \tag{2.5}$$

Volume condition presents another constraint for our deformation. We want the volume of the mesh to remain unchanged and, therefore, the formula for the closed mesh volume stands on the left-hand side of the equation 2.7 and the original volume of input mesh stands on the other side. Calculation of the mesh volume based on the vertex coordinate matrix X is the sum of partial tetrahedron volumes for N_t triangles:

$$V(X) = \frac{1}{6} \sum_{i=1}^{N_t} (\overrightarrow{x_{i_1}} \times \overrightarrow{x_{i_2}}) \cdot \overrightarrow{x_{i_3}} \tag{2.6}$$

which enables writing the volume condition in the form:

$$V(X) = \hat{v} \tag{2.7}$$

where $\hat{v}$ is the original volume of the input mesh.

We note that it is not necessary to calculate the volume on detail mesh from the beginning of the iterative deformation process since the volume of the detailed mesh can be quite well estimated from the volume of the coarse mesh. Hence, it gets useful to use the coarse mesh at the beginning to accelerate initial computations and switch to the detail mesh later.

2.3. Solution. All above stated conditions are expressed as functions of mesh vertex coordinates x_i. Using matrices, linear conditions can be merged:

$$\begin{pmatrix} \mathcal{L} \\ S \end{pmatrix} X = \begin{pmatrix} \delta(X) \\ s' \end{pmatrix} \equiv L \cdot X = b(X)$$

so that they form an over-constrained linear system with a non-linear boundary constraint of volume:

$$f(X) \equiv LX - b(X) \tag{2.8}$$

$$g(X) = V(X) - \hat{v} = 0 \tag{2.9}$$

This system can be solved iteratively using Gauss-Newton method with Lagrange coefficients. In each iteration, linear change minimizing the error of the linear Laplacian and skeleton conditions is calculated using the gradient descent formula. This change is then modified using correction directions based on the gradient of the volume function and the size of the volume error. It keeps boundary condition as hard constraint and produces vector h describing the final change. This vector is then multiplied by the desired step length α and added to the current solution to get a new position of all vertices. Formulas are:

$$\begin{aligned} &X_{k+1} = X_k + \alpha \cdot h \\ &h = -({J_f}^T J_f)^{-1}({J_f}^T f + {J_g}^T \lambda) \\ &\lambda = -(J_g({J_f}^T J_f)^{-1}{J_g}^T)^{-1}(-\tfrac{dV(X)}{\alpha} - ({J_f}^T J_f)^{-1} f(X)) \\ &dV(X) = g(X)\tfrac{g(X_0)}{g(\tilde{X}_0)} \text{ or } dV(X) = g(\tilde{X}) \text{ near last iteration} \end{aligned} \tag{2.10}$$

where J denotes Jacobi matrix of $f(X)$, $\tilde{X}$ is matrix of vertex coordinates of the fine mesh and index 0 denotes the original input state. As the result of our experiments parameter α was selected to be adaptive and to decrease from initial value 0.5 to final value 0.125. For details, see [4].

The deformation of the coarse mesh instead of the full size mesh reduces the sizes of equation matrices and improves both speed and stability of the solver. However, the usage of detail mesh for volume condition then requires vertex coordinates of the original mesh to be recalculated after each iteration. This is costly and, therefore, recommended only when the process is close to termination (i.e., $|h_i| <$ given ε, which is 0.05 in our case) so that few remaining deformations just tune the volume and do not change the shape of object much.

Iterations end when the largest change added to position of some vertex decreases under the predefined threshold.

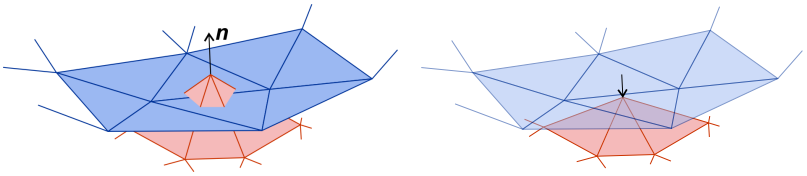
3. Deformation of multi-mesh system.

3.1. Introduction. Complex systems such as human body contain multiple meshes to be deformed. Running the algorithm described above on each mesh individually would likely cause their surface to intersect each other (see Fig. 4.1a). Correcting this problem in post-processing could cause the skeleton constraint violation and most importantly significant errors in volume preservation.

Therefore the method had to be modified to handle multiple input in the deformation phase itself and a new constraint had to be added to prevent meshes from intersecting each other. Furthermore, some other meshes may serve as rigid obstacles that may not be intersected by meshes being deformed. The obstacle meshes must enter the deformation as well, although they are not modified and they serve strictly as limitation to other meshes.

All meshes are preprocessed in the same way as in the original method. Only construction of conditions for obstacle meshes is omitted as it would be of no use. For the sake of simplicity, we will consider two meshes only, unless explicitly expressed otherwise.

3.2. Initial intersections. Input meshes are allowed to have partial intersections in the input. These are usually caused by inaccuracies in the surface mesh extraction from volume data (MRI). The partial intersection is such an intersection that every vertex can be "dragged" out of other mesh without causing other vertices to get inside the other mesh. In other words, such two meshes do not fully penetrate through their volumes (see Figure 3.1a) and the correction is reverse of the collision movement which gets the vertex out of the other mesh (see Figure 3.1b).



(a) Before fix (b) After fix

Fig. 3.1: Unregistered component of one mesh inside other. Corrected by inverted normal direction.

To remove the initial intersections, the meshes are checked in pairwise manner before the iterations of solver may begin. Every vertex of mesh is inspected using a randomly generated ray cast from its position. The number of intersections of the ray with the other mesh is calculated. If the outcome is odd, the vertex lies inside the other mesh.

To speed up the intersection detection, the space of the other mesh is subdivided using a uniform 3D grid and only those triangles lying in its cells intersected by the casted ray are precisely tested for an intersection.

If some vertex $\overrightarrow{x_i}$ is found to lie inside the other mesh, it must be moved so that the intersection is corrected. It is therefore important to find a proper direction and distance for this movement.

First, all vertices of both tested meshes that lie inside the other mesh are detected (see Fig. 3.2a). Then connected graph components of original meshes are built from these vertices (see Fig. 3.2b). These components are the regions of the surfaces that will be modified. The centre of each region is calculated as an average of its vertices.

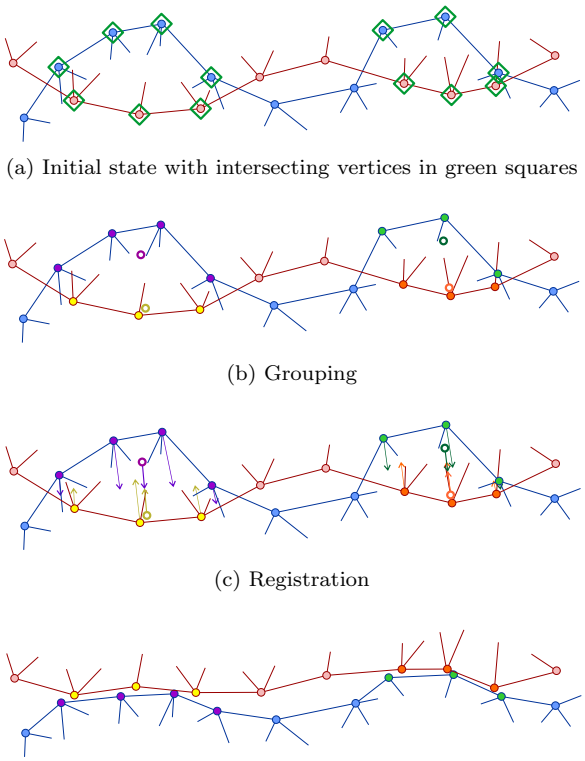
Nearest regions from both meshes are paired by selecting one region from mesh A and one region from mesh B that have minimum centre to centre distance among those yet unpaired (see fig. 3.2c).

Some maximum allowed distance must be used to prevent pairing of non-corresponding groups. In our case this is 10% of the mesh bounding sphere radius. Hence some regions may remain unpaired. Such situation can be seen in Fig. 3.1a.

If the region is paired with a region from the other mesh, then the fix direction is determined by the direction between their centres (see Fig. 3.2c) as an vector specifying the direction of translation of individual vertices in the region. Using the same direction for all vertices prevents surface triangle distortion that would be result of fixation according to local normals (see Fig. 3.3).

If one of the meshes is a hard obstacle, then all vertices of the region from the deformable mesh are moved far enough to get onto the surface of the other mesh.

However, if both meshes are deformable, then vertices from both regions are



(a) Initial state with intersecting vertices in green squares

(b) Grouping

(c) Registration

(d) After fix

Fig. 3.2: A correction of initial intersections based on mutual registration of intersected vertex groups. Correction directions stated by mutual positions of their centres (empty circles).

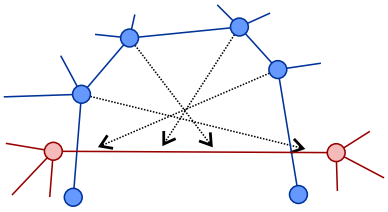


Fig. 3.3: Correction of intersection by individual vertex normal directions causing self-intersections in mesh.

moved in the opposite direction to approximately half distance to the other surface (see fig. 3.2d). Analysis showed that even though the half distance sounds like obvious choice, it is not enough for some configurations of region vertices (see Figure 3.4). To diminish this problem, value 0.6 is used. If this was too much, we might expect that condition of Laplacian will push the meshes back together to restore the original shape.

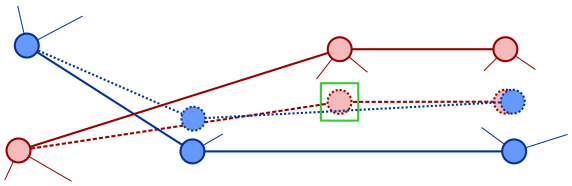


Fig. 3.4: Correction of region intersections (solid) using movement to exactly half distance to the other mesh (dashed) may not be enough to resolve the problem in some configurations (see highlighted vertex).

If the region is unpaired the fix is done as follows. Such a region is supposed to be very small, often consisting of only a very few vertices, because otherwise, there would be some vertex of the other mesh detected inside the intersection volume thus the region would be paired. From this assumption, the degenerative case (see Fig. 3.3) discussed above is limited to small fragments of the mesh and all vertices of unpaired intersection region can be simply fixed in the direction opposite to average of local normals (see fig. 3.1b). This averaging ensures that all movements are done in parallel and therefore reduces a risk of local self-intersections.

In this unpaired case, only the intersecting mesh can easily be modified to fix the intersection. Therefore, if the intersecting mesh is deformable, then full move of the vertex to the surface of the other mesh is always done to fix the intersection. If, however, the intersection vertices belong to the rigid obstacle mesh then there is no easy way to fix the intersection for now and an assumption is that such an error is small and will be eventually fixed automatically during the solution.

3.3. Solver constraint. The algorithm described in the previous section ensures that there are no significant intersections at the input of deformation iterative solver. This step is required because otherwise the following algorithm could produce intersected meshes or some unphysiological deformation.

The iteration step of the original single-mesh deformation solver must be modified. During each iteration, equations 2.10 are evaluated and new positions of vertices $\overrightarrow{x_i}$ for mesh X are obtained by addition of step vector $\overrightarrow{h_i}$. This vertex translation, however, can cause meshes to intersect.

Therefore, after calculating steps $\overrightarrow{h_i}$ for all vertices of all meshes using the original formula, we will inspect the new vertices of deformed meshes using the same test as described in the previous section.

If some intersection is found, it must be fixed. Our situation is, however, easier than previously because we know that vertex $\overrightarrow{x_i}$ was not inside of the other mesh before the application of step $\overrightarrow{h_i}$ and, therefore, we can fix the problem by a movement in the direction opposite to $\overrightarrow{h_i}$, i.e. reversing the last deformation step. This means that no grouping and pairing is involved this time and even unpaired vertices are fixed properly. Same rules as previously apply. If the other mesh is rigid, $|\overrightarrow{h_i}|$ is used as distance to get point $\overrightarrow{x_i}$ onto the surface of other mesh in direction of $-\overrightarrow{h_i}$. If both meshes are deformable, this distance is then multiplied by a safe coefficient of 0.6 causing the meshes to have some minimal space in between.

In this way the deformation is prevented to cause intersections. However, we

might expect that in the next step, the same intersection would occur as the condition of Laplacian tries to recover the local surface shape, the skeleton condition might push meshes against each other and the volume condition tries to expand the mesh volume.

We can ignore the first two conditions as they are soft and cannot be fulfilled, if a physical constraint blocks the deformation. It will cause extra local deformations of a shape as a mutual effect of two touching objects, which is consistent with the real world.

The volume constraint, however, should still be fulfilled since both objects being deformed are considered incompressible regardless of their mutual position.

If we ignored this constraint, the volume condition would cause touching area's of two meshes to expand towards each other and the intersection prevention phase would then cancel this adjustment. In theory, the volume would expand in other parts of the mesh anyway. However, it would unnecessarily slow down the process and, moreover, could cause an unresolved residuum at the end.

Therefore, we mark vertices that were found inside the other mesh in the intersection fix of the step k of iterative deformation solver. Then in the next iteration $k+1$, we modify the Jacobi matrix of mesh volume function J_g by putting zeros at corresponding rows. As a derivative function, each row of J_g describes how much respective vertex can increase the inner volume if moved properly without considering penetration of meshes.

As we know, vertices that are being found intersected are returned back. This means that they do not participate in the volume increase. So we will just put zero vector at respective row of J_g for all vertices found intersecting in the previous iteration. This will make the size of members in $J_g^T \cdot J_g$ in the equation 2.10 smaller. Consequently the volume fix part will be larger since the dot product in the denominator of the equation 2.10 for h serves as a normalisation factor. Finally, it will hint the solver to enlarge the volume more and to distribute this enlargement to other parts of the mesh.

This modification of J_g, however, does not prevent intersection in the next iteration as there are still other two conditions left that may move the problematic vertex to wrong direction. We cannot cancel those and, therefore, the intersection prevention mechanism must be run after each iteration.

Although described for two meshes only, this approach can be used for more inputs by a consequent application in pair-wise manner. It is important to do it sequentially as fixing intersection of mesh A with mesh B might change the state of intersection of mesh A and C, thus, if both fixes of A versus B and A versus C were applied at the same moment, the total adjustment could be excessive and cause unnecessary deformation of one or more meshes. To prevent this, intersection of mesh A and B is solved first and resulting meshes A' and B' are then inputs of A' versus C fix producing C' for later B' versus C' test.

It is easy to see that not all meshes are typically colliding. Therefore precise investigation of their intersections is unnecessary and should be avoided. We use capsule shaped bounding objects oriented according to the main axes of meshes that are found using the principal component analysis (PCA) of their vertices. The intersection fix step is then only run for pairs with intersected bounding objects.

All intersection tests are run with coarse meshes as well as the deformation itself. As the coarse mesh used by the method is supposed to be an outer hull, if two coarse meshes do not intersect, then the same applies to their corresponding original meshes. If the coarse mesh is not guaranteed to be an outer hull, one final intersection

test and possibly fix must run on detail models. It is important to note that this causes significant slowdown of the process as the intersection detection complexity is quadratic in mesh vertex count. Furthermore, it can also grow the volume preservation error.

4. Results.

4.1. Comparison with the unmodified algorithm. Our approach was implemented in C++ (MS Visual Studio 2010) under the Multimod Application Framework – MAF [6], which is a visualisation system based mainly on VTK [5] and other specialised libraries. Our implementation was then integrated into the MuscleWrapping software [1] that is being developed within the VPHOP project [7]. The algorithm was tested on medical data in our human body framework. The meshes of muscles were used as deformable objects while mesh of bone served as a rigid obstacle.

An example of our results can be seen in Fig. 4.1. There was the total number of 3 meshes with average number of 4 400 vertices involved in the test. While Fig. 4.1a shows intersected regions when no prevention mechanism was used, Fig. 4.1b shows the same view after the application of the described modification. As it can be seen, intersections were avoided by the algorithm. The downside of the modification is a longer execution time. The iteration count has risen from 113 to 200 as a result of the intersection preventing constraint added to the solver. The intersection detection tests then caused additional slow down so the final deformation time has risen from 2 677 ms to 6 983 ms on PC with Core i7 2600K at 3500 MHz. This is lower than could be expected from finite element based methods and suitable for preliminary judgement of the patient condition and necessity of a more precise examination. Keeping the time in range of seconds per frame is vital when applied to multi-frame animation.

Another drawback of the approach is a larger error in volume preservation. The unmodified solver was able to maintain the original volume with ratio 0.99260, while the extended version achieved ratio of 0.96365 only. This is however sufficient for our application as volume loss up to 6% was stated to be physically acceptable [1].

Anyway, we believe that the new algorithm produced more realistic results since intersections of meshes stand for a more serious problem than accuracy of volume preservation.

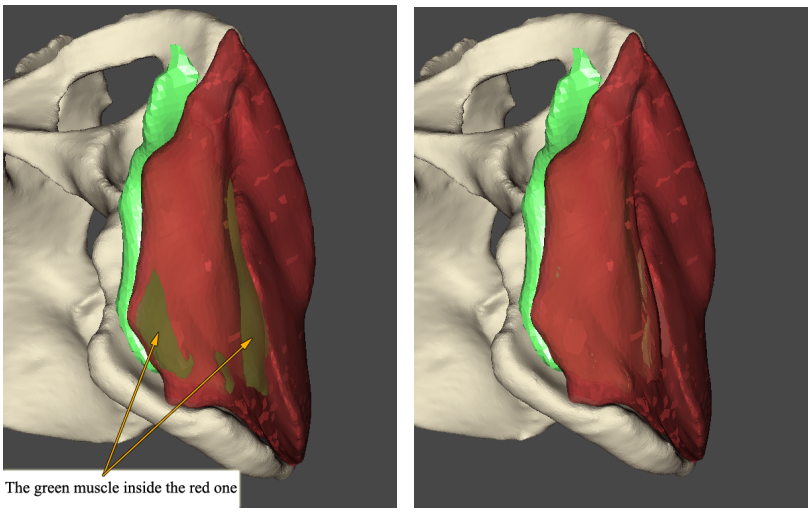
4.2. Remaining issues. The proposed method works acceptably provided that initial conditions are fulfilled. This means that both the input meshes and their coarse meshes can only have a limited extent of initial intersections. Unfortunately, in some real cases, as we have recently found, this condition is not fulfilled. An example can be seen if several muscles attach to the same area on the bone, e.g. Semitendinosus and Biceps Femoris on Pelvis. Initial intersection fix directions are stated wrongly in such configuration and the result of deformation is distorted.

Correcting this issue is a matter of further investigation. Packing the starting meshes to elementary volumes with natural expansion using linear conditions during solving was tested but this did not prove to be reliable as intersection often prevented the meshes to fully expand to their original shape and size.

5. Conclusion.

In this paper, we described the modification of original method for simultaneous skeleton driven deformation of multiple interacting meshes with inner volume preservation and mutual intersection prevention. Tests show that the method

[1] http://graphics.zcu.cz/Projects/Muskuloskeletal-Modeling



(a) Original method (b) Intersection prevention

Fig. 4.1: Deformation of Gluteus Minimus (green) and Gluetus Medius (red, transparent) constraint by Pelvis (white). Comparison of the original method and the modified method. The highlighted areas of intersection denotes surface of G. Min. inside the volume of transparent G. Med., not full penetration.

works as expected, if the assumption about limited initial intersections is fulfilled. The result are deformed meshes with no or nearly no intersections and the same volume as before.

Acknowledgments. This work was supported by the Information Society Technologies Programme of the European Commission under the project VPHOP (FP7 ICT-223865) and by the Ministry of Education of The Czech Republic under the project 7E11016.

References.

[1] Amaury Aubel and Daniel Thalmann. Efficient muscle shape deformation. In *in Deformable Avatars , IFIP TC5/WG5.10 DEFORM2000 Workshop*. Kluwer, 2000.

[2] Jin Huang, Xiaohan Shi, Xinguo Liu, Kun Zhou, Li-Yi Wei, Shang-Hua Teng, Hujun Bao, Baining Guo, and Heung-Yeung Shum. Subspace gradient domain mesh deformation. *ACM Trans. Graph.*, 25(3):1126–1134, 2006. URL `http://dblp.uni-trier.de/db/journals/tog/tog25.html#HuangSLZWTBGS06`.

[3] Tao Ju, Scott Schaefer, and Joe D. Warren. Mean value coordinates for closed triangular meshes. *ACM Trans. Graph.*, 24(3):561–566, 2005. URL `http://dblp.uni-trier.de/db/journals/tog/tog24.html#JuSW05`.

[4] Josef Kohout, Petr Kellnhofer, and Saulo Martelli. Fast deformation for modelling of musculoskeletal system. In *Proceedings of the International Conference on Computer Graphics Theory and Applications: GRAPP 2012*, pages 16–25, Rome, February 2012.

[5] Will Schroeder, Ken Martin, and Bill Lorensen. *The Visualization*

Toolkit, Third Edition. Kitware Inc., 2004. ISBN 1930934122. URL http://www.worldcat.org/isbn/1930934122.

[6] Marco Viceconti, Luca Astolfi, Alberto Leardini, Silvano Imboden, Marco Petrone, Paolo Quadrani, Fulvia Taddei, Debora Testi, and Cinzia Zannoni. The multimod application framework. *Information Visualisation, International Conference on*, 0:15–20, 2004. ISSN 1093-9547. URL http://doi.ieeecomputersociety.org/10.1109/IV.2004.1320119.

[7] VPHOP. the osteoporotic virtual physiological human, http://vphop.eu, 2010.