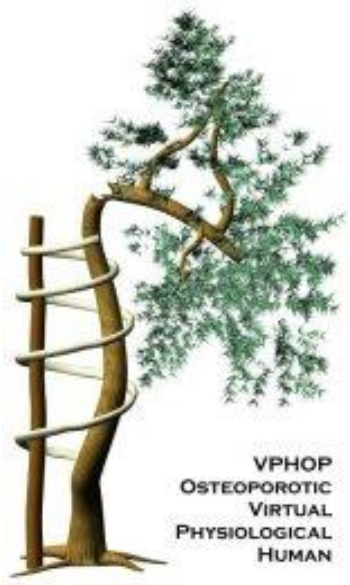


VPHOP

Technology to fight Osteoporosis

Integrated Project # FP7-223865

User Manual

Morphable Musculoskeletal Model

Work package WP10: Morphable Musculoskeletal Model

Josef Kohout – (UWB)

Gordon J. Clapworthy – (BED)

5.8.2012

TABLE OF CONTENTS

1. Introduction

An estimation of skeletal loading is feasible only through a modelling technique [1]. Musculoskeletal models [2]-[6] in common clinical use assume that the mechanical action of the muscle occurs along a poly-line, namely the action line, joining the origin and insertion points of the muscle, i.e., the sites at which the muscle is attached to the bone by a tendon. In essence, an action line is a representation of muscle fibres. An advantage of action-line models, besides their rapid processing speed, is that the model created for one particular patient can be easily adjusted (usually by uniform scaling) to another patient, though, of course, if the anatomies of the patients differ significantly, this may no longer be true. On the other hand, these models tend to overestimate the predicted joint loads because their assumption of muscle fibre length being uniform over the entire muscle bundle is often not fulfilled in practice [1]. Another drawback is that representing a muscle by a set of *ad hoc* action lines provides very limited insight since it is so visually different from the reality of human anatomy.

More accurate approaches represent a muscle by a B-spline solid whose iso-lines correspond to muscle fibres [7] or by a 3D finite-element mesh whose cells contain information about the direction of the muscle fibres present in its volume [8]. Although good agreement was found when comparing the results with static MRI images taken in different postures, use of these models in the clinical context is highly impractical because generating the meshes is a complex process easily requiring several days for a highly skilled operator and the model cannot be easily adjusted to another patient, and, once the model is generated, computing the solution requires several hours on a supercomputer [8].

Our approach attempts to achieve the advantages of both strategies. We start from a generic atlas model, in which bones and muscles are represented by their triangulated surface meshes in the rest-pose position, and then deform this according to data captured from the individual patient to form a fully personalised model. This is done in three stages:

- *atlas scaling*: the generic atlas model is semi-automatically scaled non-uniformly (morphed) to fit the anatomy of a particular patient defined from EOS dual images,
- *motion data fusion*: the morphed atlas model is fused with motion data defining the kinematics of the skeleton during various physical activities,
- *muscle wrapping*: for each time frame (current-pose position), the positions and shapes of the muscles are calculated; interpenetration is avoided so that muscles wrap properly around the bones and other muscles, and a muscle fibre model is accommodated within the deformed muscle.

This document describes all these three stages (from the user point of view) as they were implemented and integrated into LHPBuilder software[1].

[1] The user manual of LHPBuilder, including description of system requirements, can be accessed from https://www.biomedtown.org/biomed_town/LHDL/users/swclient/UserManual

2. Installation

The easiest option is to download the latest version of LHPBuilder installer in the standard way and, once it has been downloaded, run the installer. Should you experience any trouble with the latest version, please, download WP10 release version of LHPBuilder from:

https://www.biomedtown.org/biomed_town/vphop/consortium/wp10/repository/Tools.

You should also download the latest version of the "Generic_Atlas_Model_WP10" data set (unless you are an experienced user and want to create your own generic atlas model) from:

https://www.biomedtown.org/biomed_town/vphop/consortium/wp10/repository/Data.

Four different data sets are present in the repository.

- *Generic_Atlas_Model_WP10_vX.7z* is the default data set, which is supposed to be used in clinical practice; it contains nothing other than the required generic atlas model.
- *Generic_Atlas_Model_WP10_vX_DEMO.*7z contains a model that was fused by walking motion data; this data set is supposed to be used for demonstrations.

The other two files contain data to be used for testing – it is to be emphasised that THE FILES BELOW SHOULD NOT BE USED FOR ANY PURPOSE OTHER THAN TO TEST THE WP10 MORPHABLE MUSCULOSKELETAL MODEL

- *Generic_Atlas_Model_WP10_vX_TEST_ONLY.7z* contains some predefined motion capture data and EOS patient image data to quickly test the functionality.
- *Generic_Atlas_Model_WP10_vX_TEST_ONLY_FULL.7z* extends the previous data set by including walking model and landmarks for atlas scaling.

The data sets are compressed using the LZMA2 compression technique to be found in 7-Zip (http://www.7-zip.org/), so after downloading the required data set, all the files contained must be extracted to some folder.

3. Getting Started

Run LHPBuilder, log in (this may be skipped since it is not needed for WP10), and open the downloaded *Generic Atlas Model*. This should produce a display similar to Fig. 3.1.

Each data entity is represented by a VME. The VME is organised into a hierarchical tree (VME Tree or data Tree) and consists of: a time-varying dataset, a time-varying matrix (which defines the pose of the VME relative to its parent in the VME Tree), and a set of metadata (which provides all of the textual attributes of the VME).

The Musculoskeletal Model (either generic, scaled or fused with motion data) is represented by a hierarchy of various VMEs rooted with a *medVMEMusculoSkeletalModel* VME. In the data loaded, such a VME can be found under the name *Atlas Model*, and it represents the static, incomplete generic atlas model of the lower limbs.

Besides its hierarchical functionality, this root VME provides the user with a GUI to set up the muscle wrapping – see Section 6. Under this root, one can find regions, joints (which interconnect different regions), wrappers (i.e., traditional action lines) and, most importantly, muscle wrapper VMEs which provide the user with the surface and muscle-fibre representations of a muscle wrapped around bones.

Each region, which represents one part of the human body (and its visual representation, therefore, should be the skin of that part), contains one *mafVMELandmarkCloud* VME named *MotionLs* which contains landmarks placed on the surface of this region. These landmarks are used in the motion fusion stage (see Section 5) to find the proper rigid transformation of its parent region. We note that, currently, our generic atlas model is incomplete and thus some muscles are missing.

Each region also contain *Bones* and *Muscles* groups; the former includes surface models of the bones at various levels of detail and the muscle attachment areas, the latter surface models of muscles (again at various level of detail) and, optionally, muscle fibre geometry extracted from an autopsy for the same human body. Also present may be hulls (i.e. exterior bounding surfaces at the various resolu-

tions), which are generated automatically during the muscle wrapping stage (see Section 6) and stored to speed up the process.

If you want to learn how to modify the atlas (or even to create your own), please see Section 7.

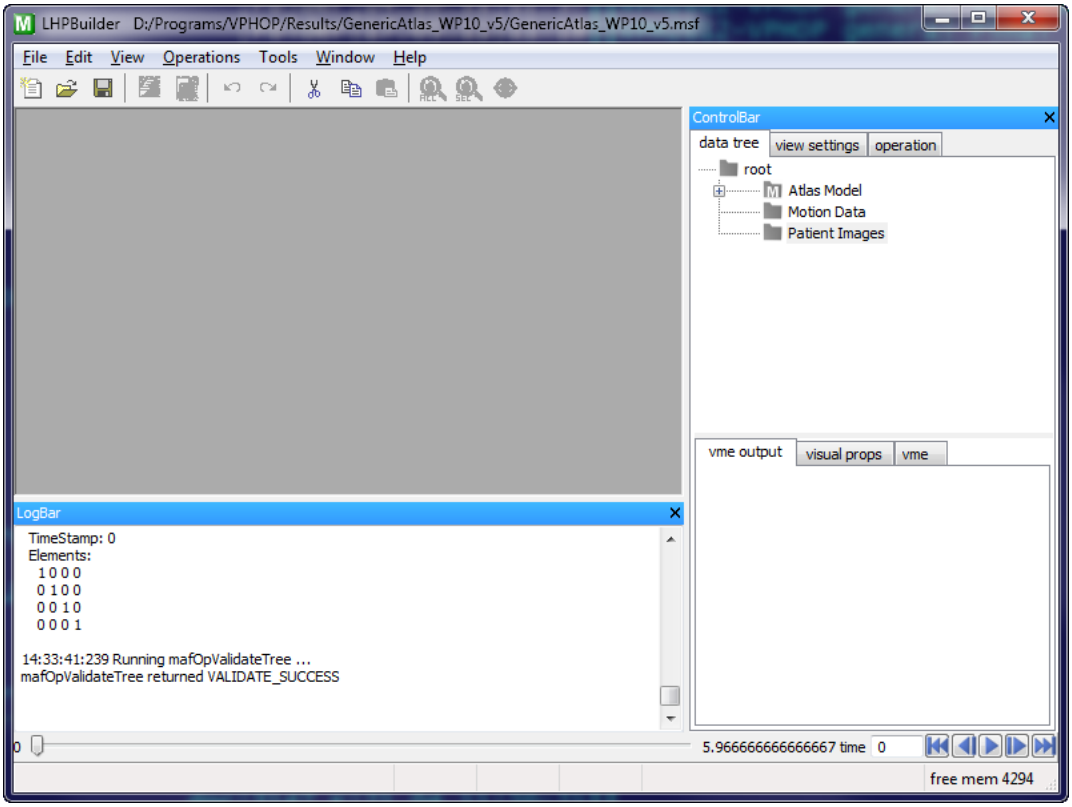


Fig. 3.1. LHPBuilder

The following sections describe various stages of our morphable musculoskeletal model. Each stage is fully independent of the others, though best performed in a systematic way, i.e., you may proceed with any stage you want and obtain some results, though if you do so, the reliability may be low.

4. Atlas Scaling

In this stage, the generic model is scaled to fit the anatomy of a particular patient defined from EOS dual images. The workflow for this is shown in *Fig. 4.1*; we now describe it in detail.

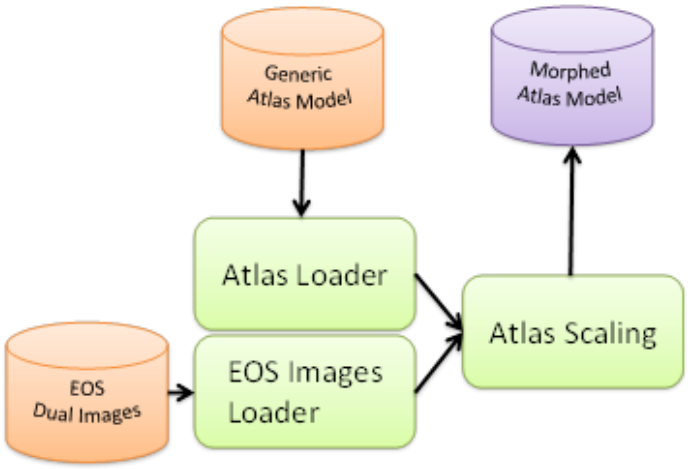


Fig. 4.1. Workflow of Atlas Scaling.

First, a set of orthogonal DXA or X-ray patient images (produced, for example, by the EOS device) is imported. It is recommended that both imported images are stored under a common *mafVMEGroup* VME to make navigation in the data tree easier – see Fig. 4.2. We note that group VMEs can be created using a standard operation accessible from the menu Operations>Create>New>Group.

Following that, an operator specifies several landmarks on both images, which is performed using the operation *EOS Landmark Extraction* which can be called from the menu Operations>VPHOP WP10 – TESTING>Create [WP10]>EOS Landmark Extraction (see Fig. 4.3). The operation itself can be run only after a group VME in the data tree has been selected. The result of this operation will then be stored under this group. Once again, it is recommended that the user carefully selects the group under which both images are to be stored (e.g., series_DX_0_1763x8417x2 images in Fig. 4.2).

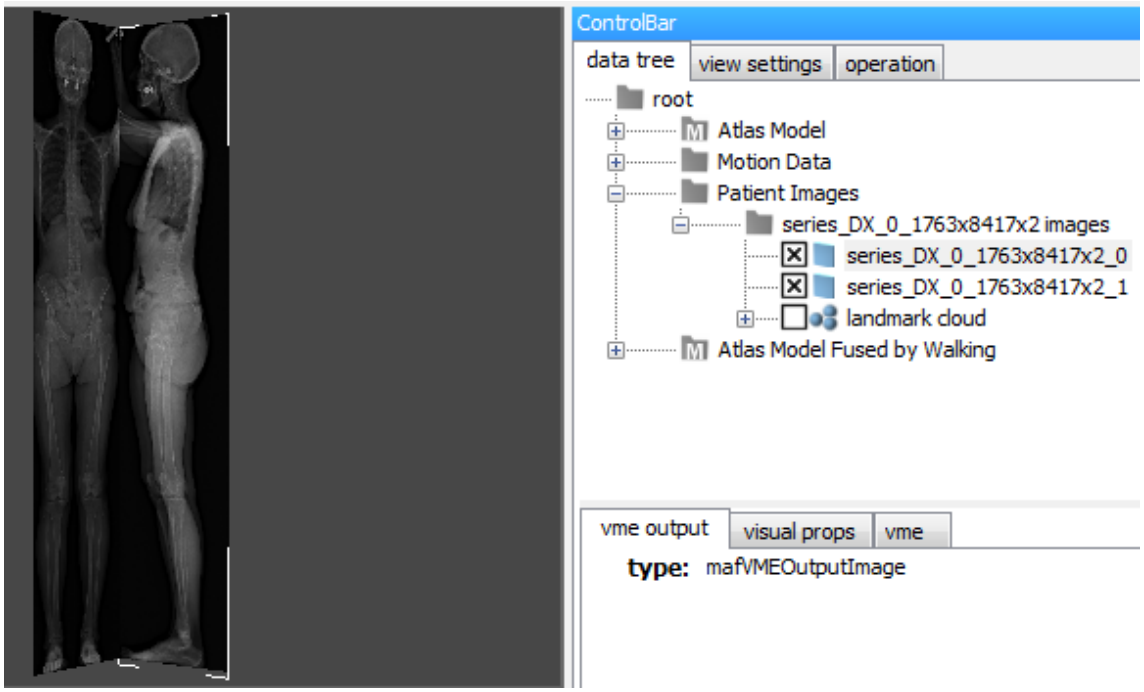


Fig. 4.2. Example of DXA patient images.

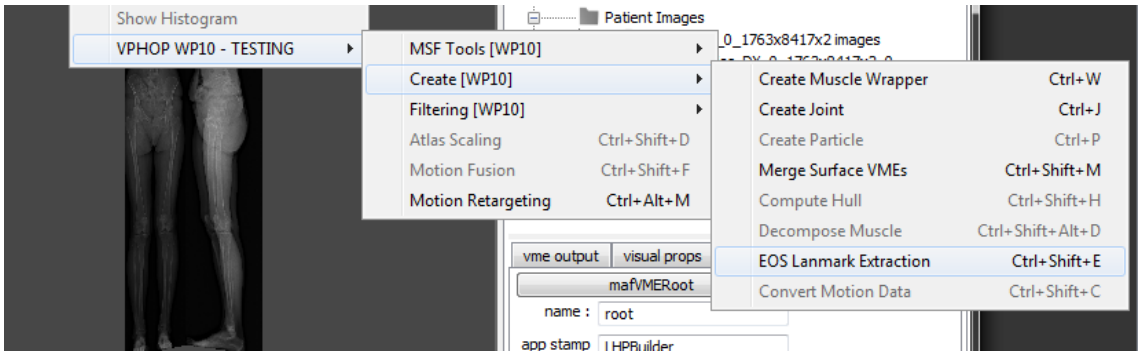


Fig. 4.3. Running EOS Landmark Extraction

In the operation GUI (see Fig. 4.4), select VMEs with DXA images (Coronal and Sagittal) and then, using the mouse, manually drag & drop 6 landmarks (see the pink balls), representing the centroids of SubTalar (ankle), FemoroTibial (knee) and CoxoFemoral (hip) joints, to their anatomically correct positions. Press OK, when you are finished. The operation automatically creates a new *mafVMELandmarkcloud* VME named "landmark cloud" with these landmarks as a child of the selected group VME upon which the operation was executed.

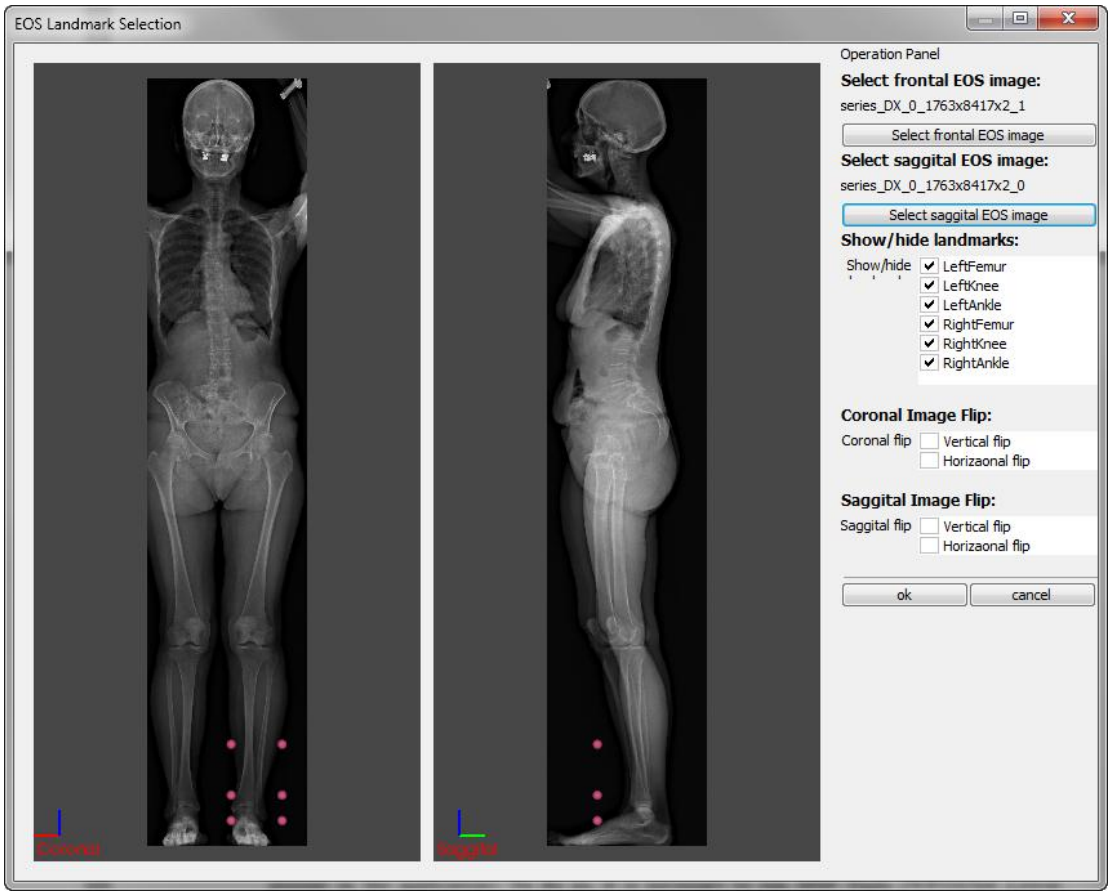


Fig. 4.4. EOS Landmark Extraction.

This landmark cloud is required for another operation, *Atlas Scaling*, which must be run on a musculo-skeletal model VME (i.e., the VME must be selected before the operation is executed). Typically, this is the static generic atlas model (see *Atlas Model* in the test data). In the operation GUI, shown in *Fig. 4.5*, select the landmark cloud created (button: *choose patient landmark*) and press OK.

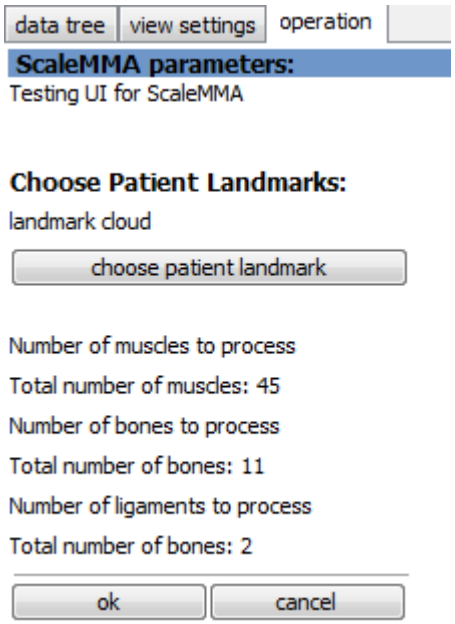


Fig. 4.5. Atlas Scaling operation

After pressing *OK*, the *Atlas Scaling* operation (also known as *ScaleMMA*) automatically takes the generic musculoskeletal atlas that has been input and scales it, non-uniformly, to a patient-specific model by using total energy minimization. Local surface details are preserved by the differential Laplacian shape descriptor while the landmark points are moved close to their target locations. Constraints of the tendon attachments and the contiguity of bone or muscle are included to retain musculoskeletal structures and avoid inter-penetration.

> **CAUTION:** As the atlas model contains dozens of high resolution surfaces of muscles and bones, this operation requires lot of memory. To reduce the possibility of the operation crashing due to lack of memory, it is recommended that, before you start running it: save everything, restart the LHP-Builder application, open the saved state, and then proceed immediately with the operation. The computation may take up to 20 minutes depending on the power of your computer, so please be patient.

Chyba! Nenalezen zdroj odkazů. shows a visualization of the most important VMEs of the generic atlas and patient-specific models.

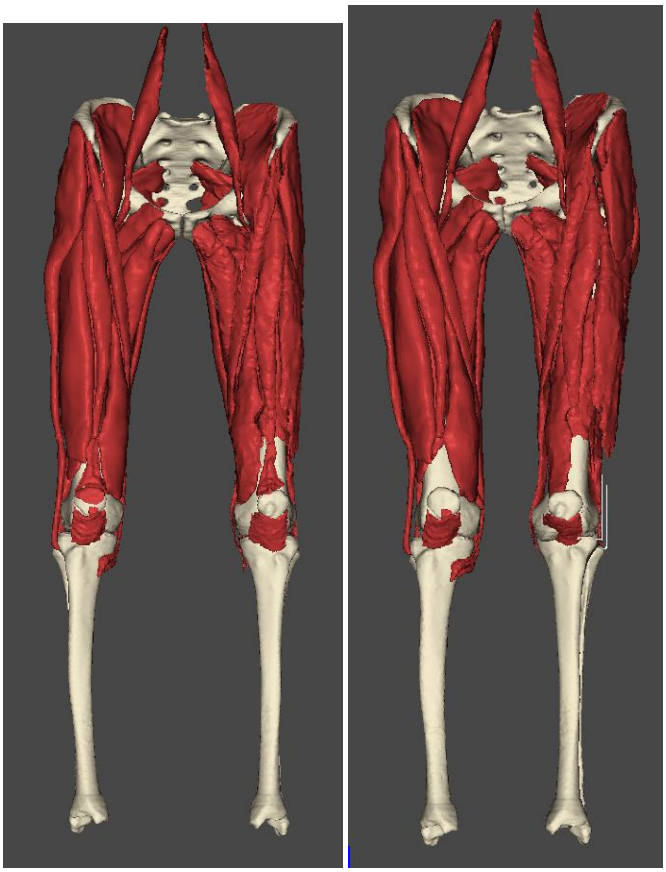


Fig. 4.6. Musculoskeletal model before (left) and after (right) atlas scaling.

5. Motion Fusion

This stage fuses the musculoskeletal model (either the generic model or that already scaled to fit the patient anatomy) with motion data defining the kinematics of the skeleton during various physical activities, e.g., walking, stair climbing or falling to one side. The workflow of this stage is given in Fig. 5.1. As can be seen, there are two alternative paths through this workflow. We shall describe each of them in detail below.

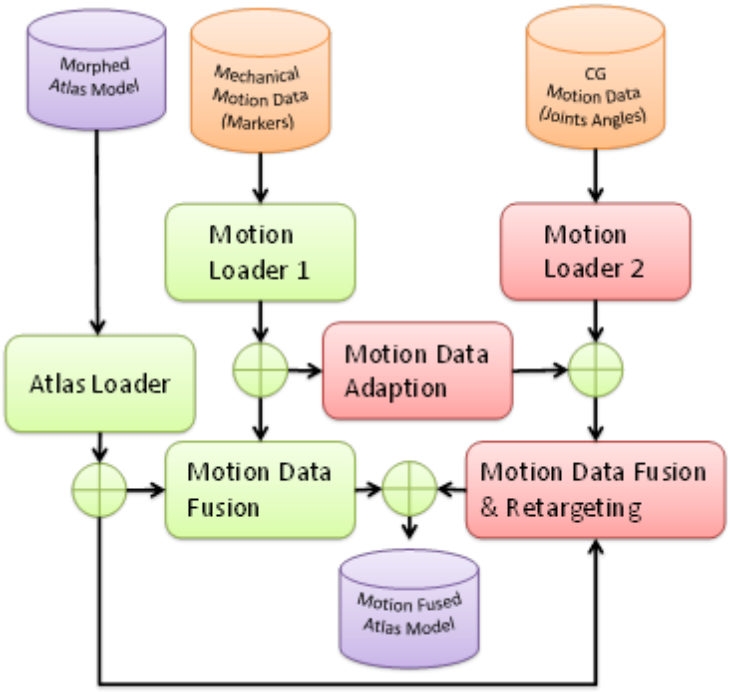


Fig. 5.1. Workflow of Motion Fusion. The green spheres indicate where more than one option exists (only one path is to be chosen).

5.1. Simple Motion Data Fusion

In this pathway, the motion data must be represented by a time-variant landmark cloud (this can be imported using traditional motion analysis importers) with landmarks of names corresponding to those present in the MotionLs landmark clouds in the atlas – see also Fig. 5.2. It is important that the motion data comes from a subject with an anatomy with similar dimensions to the atlas. If not, bones might move unrealistically (e.g., penetrate each other).

This option is most suited to cases in which we have EOS images (to scale the generic atlas model) and motion data from the same patient. We note that the *Walking* motion data in the test generic atlas model is compatible with the anatomy captured in the generic atlas model.

To fuse the atlas with such motion data, it is necessary to select the atlas and run the operation *Motion Fusion*. The iterative closest point (ICP) algorithm is used to register the time-variant landmarks from the motion data and corresponding landmarks that are specified in the model. For each time frame, the ICP algorithm translates and rotates the bones according to the input motion data, which changes the coordinates of the landmarks for the attachment areas and the end points of the action lines since the positions of these depend upon the position of the corresponding bone. We note that the *Motion Fusion* operation takes a couple of minutes (depending on the complexity of atlas model and mainly on the number of frames present in the motion data) and creates a new musculoskeletal atlas.

CAUTION: You should not delete the original static musculoskeletal atlas after the motion fusion is complete because it will be required later in muscle wrapping stage.

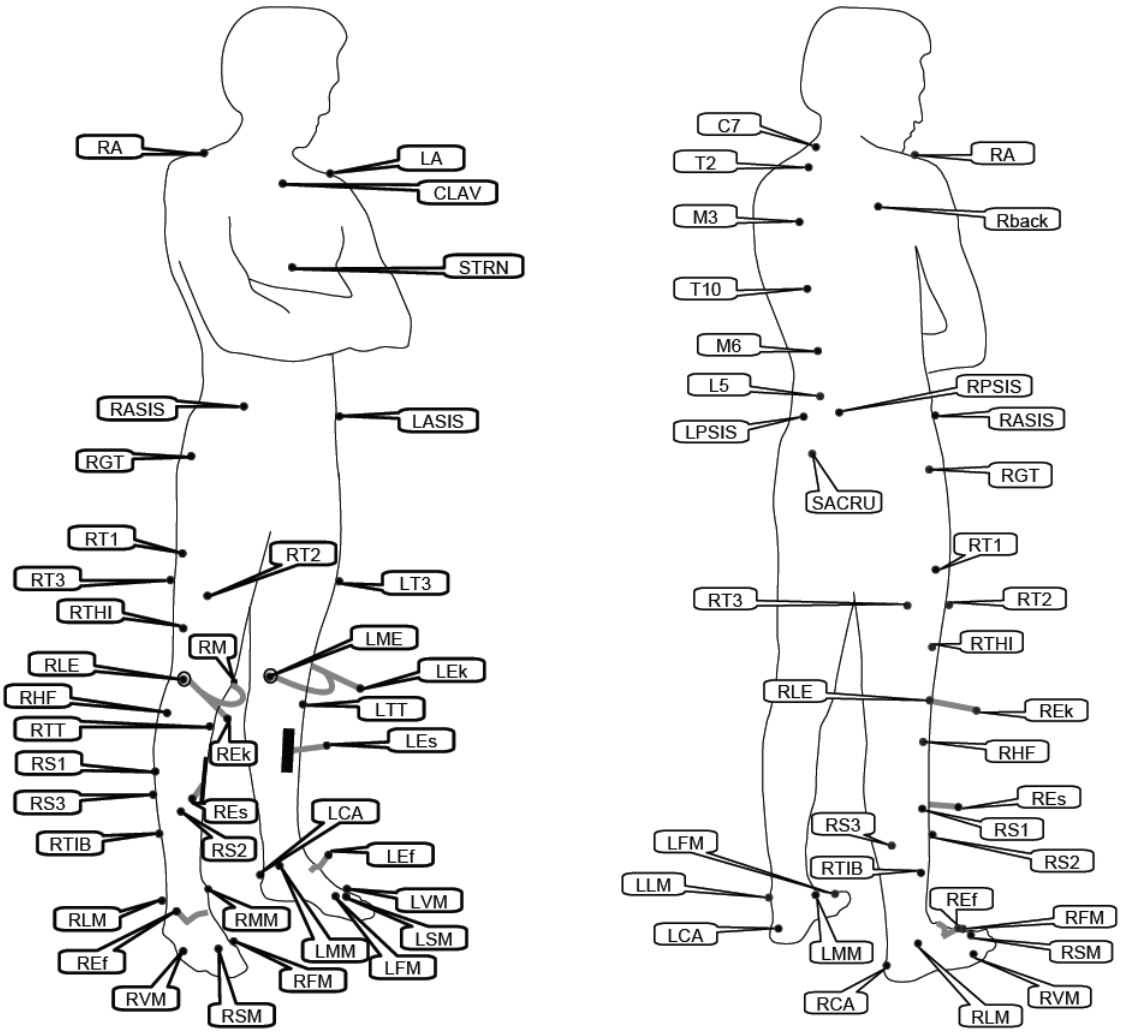


Fig. 5.2. Body Markers (image taken from [9])

5.2. Motion Data Fusion with Motion Retargeting

In this pathway, the motion data must be represented by a hierarchy of joints with time-varying positions and orientations. It can be created from a time-varying landmark cloud (see the next section) or be given in the ASF/AMC format, commonly used in computer graphics applications. While it is helpful if the anatomy of the subject is close to that of the virtual body in the model, this is not essential – the motion retargeting uses a Kálmán-like filter which is applied automatically to produce a physically plausible motion that preserves the desirable properties of the motion data in the moving object.

To fuse the atlas with such motion data, select the atlas and run the operation *Motion Fusion*, specifying the motion data as input. This operation takes a couple of minutes (depending on the complexity of atlas model and the duration of the motion); it creates a new musculoskeletal atlas in which the bones move realistically in accordance with the motion.

CAUTION. You should not delete the original static musculoskeletal atlas after the motion fusion is complete because it will be required later in the muscle wrapping stage.

5.2.1. Motion Data Conversion

Motion data represented by a time-varying landmark cloud (which has landmarks with corresponding names) can be converted into a hierarchy of joints using the operation *Create [WP10]/Convert Motion Data*. This operation must be run on the landmark cloud to be converted, and the user is asked to specify the generic atlas VME. The operation copies the joints present in the specified atlas VME and the

regions that interconnect with their MotionLs landmark clouds into a new musculoskeletal model. The operation then automatically estimates the positions of the joints from the input motion data (you may store them into the landmark cloud for a visual check) and uses these estimations, together with landmarks in the motion data, to translate, rotate and scale the regions appropriately to define the new positions and orientations of the joints.

The operation may take easily 20 minutes since the estimation of joints is based on the time-consuming, non-linear optimisation of an objective function of variables which number 3 × the number of frames.

CAUTION. Regions are used only to define joints, so, they may be unrealistically deformed, as can be seen in Fig. 5.3. Motion Fusion is required to produce an expected behaviour.

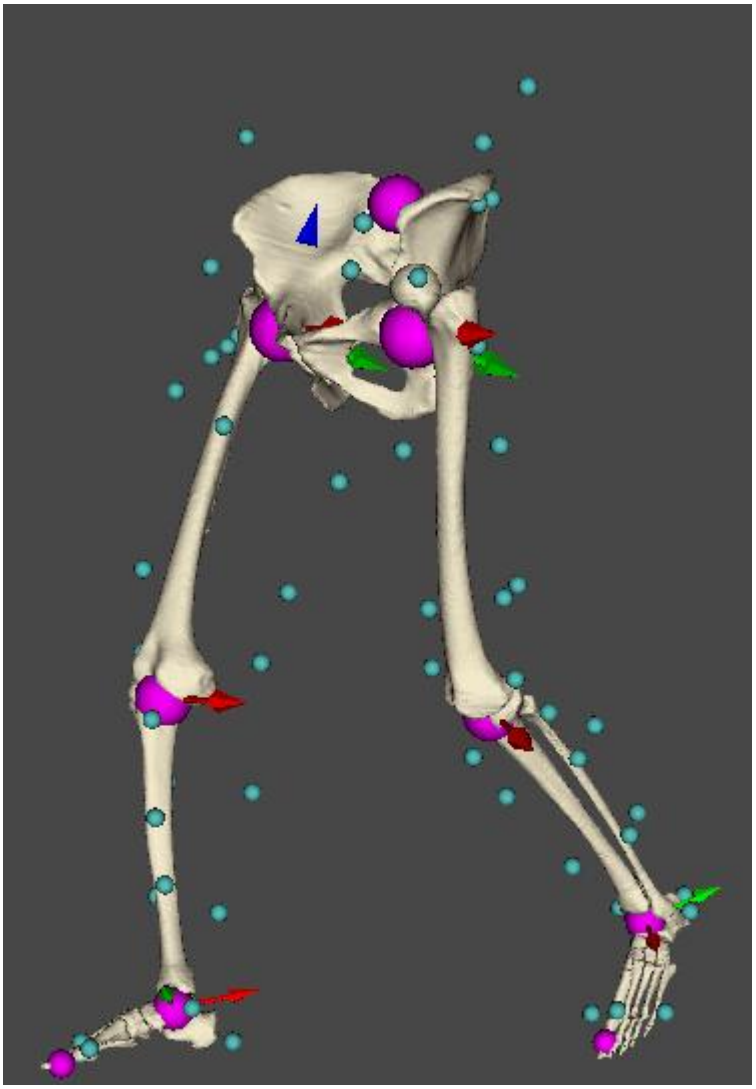


Fig. 5.3. The result of converting time-varying landmark cloud motion data (cyan balls) into a hierarchy of time-varying oriented joints (magenta balls with axes).

6. Muscle Wrapping

In this stage, the surface representation of the muscles and their fibres is wrapped around the bones as they move. Its workflow is given in Fig. 5.1; as can be seen, there are three different pathways, and each employs a different approach. The pathway to be followed can be configured in the settings of motion fused musculoskeletal atlas VME (see Fig. 6.2.) as *Deformation method*. Once the settings are changed, you must click the *Commit* button to apply the changes.

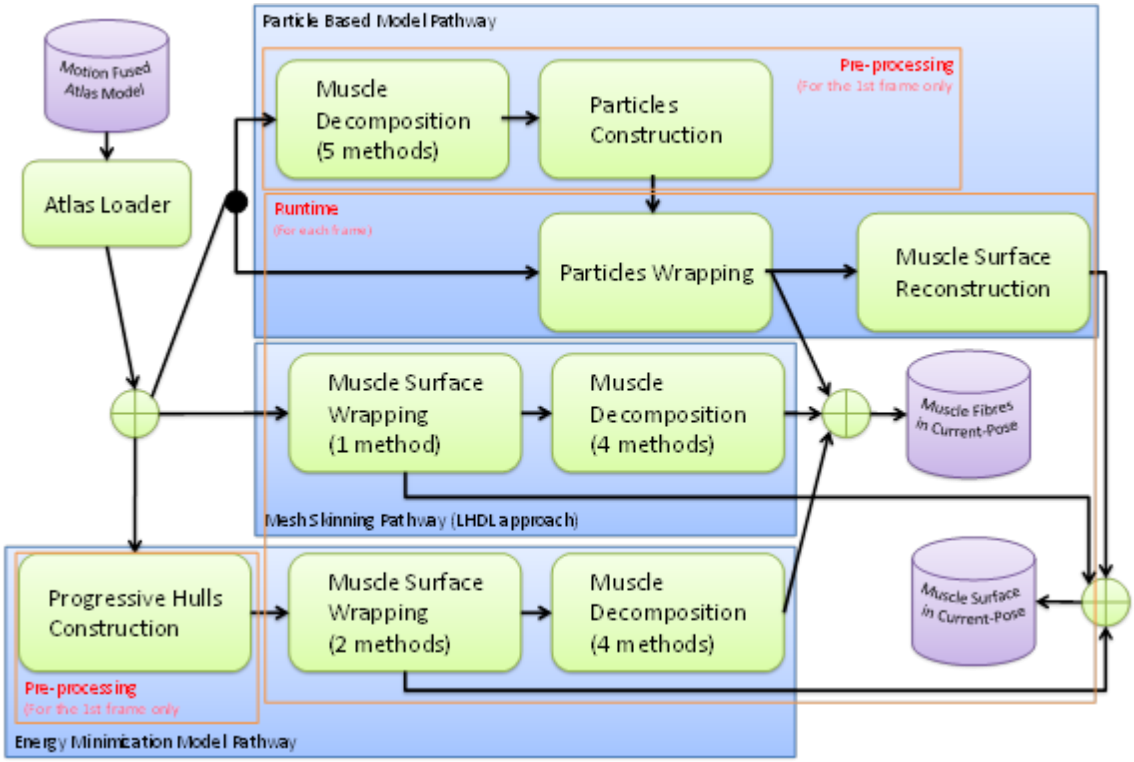
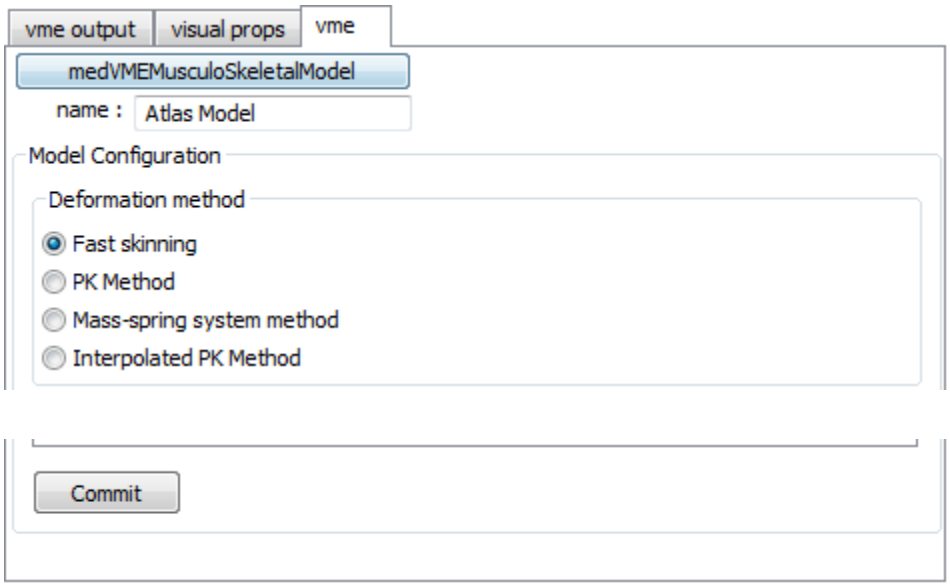


Fig. 6.1. Workflow of Motion Wrapping. Green spheres are used at places where more options exist (only one path is to be chosen).



…

Fig. 6.2. medVMEMusculoskeletalModel VME main part of GUI

For each muscle to be wrapped, one muscle-wrapper VME (*medVMEMuscleWrapper*) should be constructed and associated with the rest-pose (static) surface muscle VME from the generic atlas model – see Fig. 6.3. Muscle-wrapper VMEs observe various events including also the change of the simulation time (see the timebar in the lower part of the application window) and, when needed, they perform the wrapping of either the surface or the muscle fibres (via the checkbox *Generate fibres*). Detecting any possible change (e.g., smoothing of muscle, adding a new via point for an action line, or transforming the bone along which the muscle run) is time consuming, so by default, wrappers detect only typical changes (e.g., a change of simulation time). To enable full detection, uncheck the *Fast Checks* checkbox.

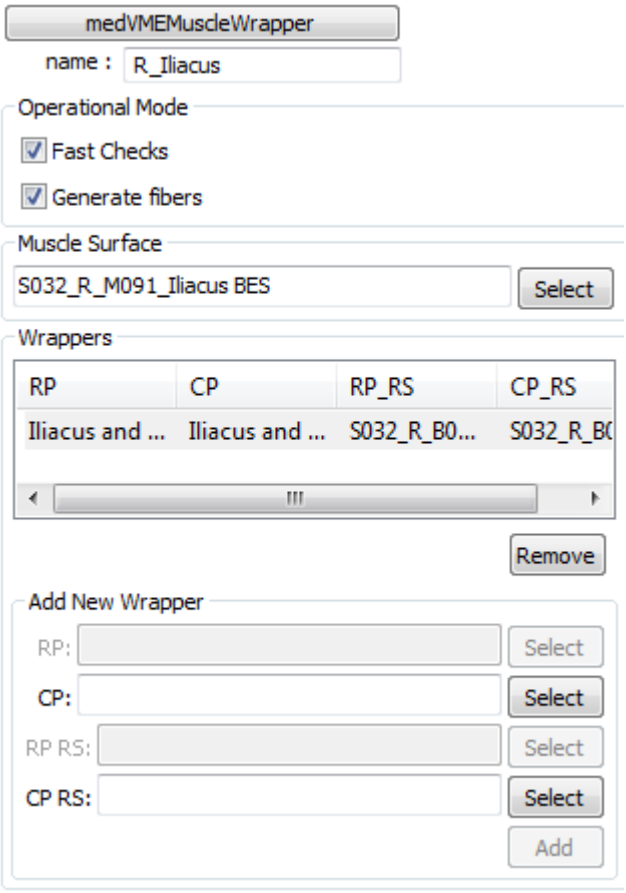


Fig. 6.3. medVMEWrapper VME GUI

We now describe each of pathways and the methods they employ, in turn.

6.1. Mesh Skinning Pathway

In this pathway, originally developed in the LHDL project, surface models of muscles are wrapped around the bones using the fast skinning deformation method (see Section 6.4.1) and afterwards decomposed using any of the existing methods for muscle decomposition apart from the *Update Particle* and *Cadaver Fibres* methods. The surface models are wrapped independently of each other, not even taking bones into account, so a wrapped muscle may intersect another muscle of even bone. The likelihood of intersection depends on the quality of the action lines (see Section 6.4.1). The main advantage of this pathway is its speed.

To enable this pathway, choose the *Fast skinning* deformation method in the settings of the motion fused musculoskeletal model VME (see Fig 6.4). All other options in these settings are then ignored.

6.2. Energy Minimization Pathway

In this pathway, surface models of muscles are wrapped around the bones using either the PK or the Interpolated PK method and are afterwards decomposed using any of the existing methods for muscle decomposition, apart from the *Update Particle* and *Cadaver Fibres* methods. Compared to the Mesh Skinning pathway (Section 6.1), the volume of a muscle is better preserved during the wrapping and also penetration between muscles and bones can be avoided. However, the current implementation may require hours to process the whole atlas; this time will reduce as current improvements to eliminate identified bottlenecks are implemented, but the computation time will remain significantly longer than for Mesh Skinning.

To enable this pathway, choose either *PK Method* or *Interpolated PK Method* in the settings of the motion fused musculoskeletal model VME. This can be seen Figure 6.4, which also shows other important settings.

It is recommended that the *Use Progressive Hulls* checkbox is enabled because this increases the numerical robustness of the chosen surface wrapping method making it less sensitive to artefacts that may be present in the models of the muscles and bones. However, construction of progressive hulls is time consuming, so provided that models in the atlas are of an excellent quality, it may be useful to disable this option for the *Interpolated PK Method*.

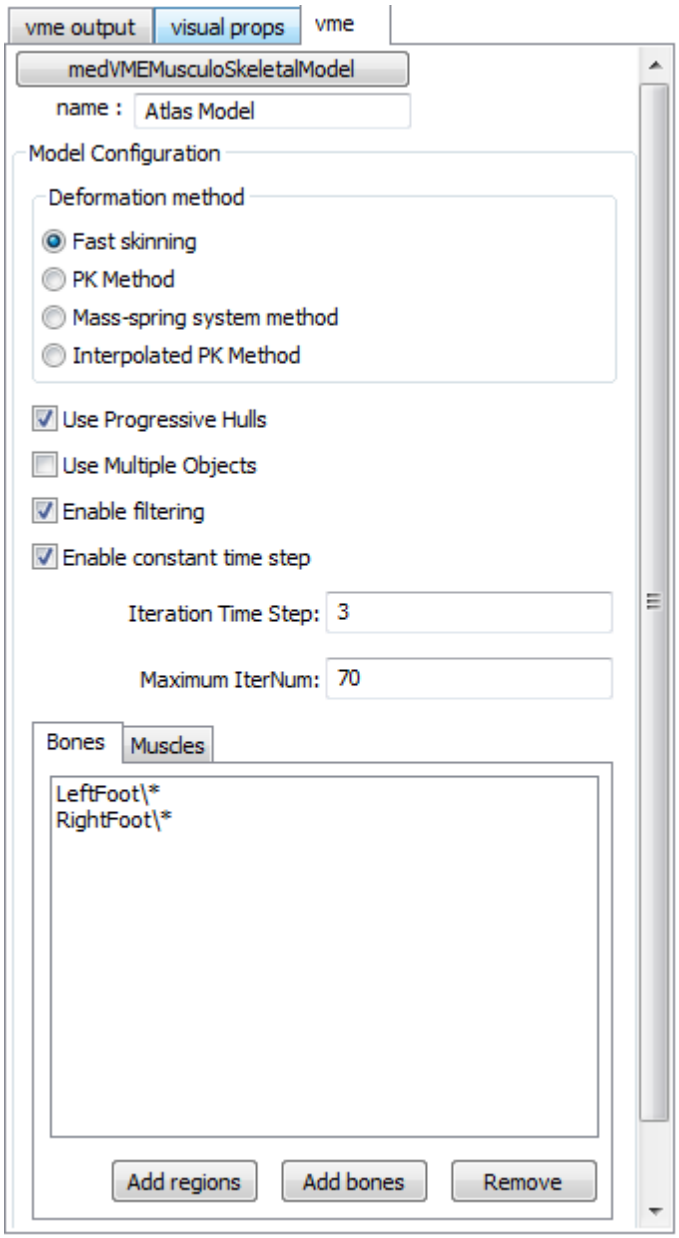


Fig. 6.4. medVMEMusculoskeletalModel VME settings

The *Use Multiple Objects* check box turns on/off the mechanism to prevent penetration between muscles and bones. When this is turned off, the muscles are wrapped independently, as in the Mesh Skinning Pathway, but an advantage of this pathway is that the methods it employs better preserve the volume of muscles during wrapping. If penetration prevention is turned off, the benefit of the *Interpolated PK* method is lost, and since it is significantly slower than its counterpart, its use in these circumstances is not recommended.

When penetration prevention is turned on, both methods for surface wrapping require several hours to process the whole atlas because of the inefficiency of current implementation, as mentioned above. In this case, it may be useful to ignore some bones or muscles to reduce the amount of penetration checking to be performed. For example, one may choose to ignore the bones of the left leg, when the right

leg is under inspection, as it is less likely that inter-penetration of these will occur. To do so, the option *Enable filtering* must be checked and the models to be ignored must be specified in the list at the end of the settings. Note: the GUI will have to be refactored when WP10 migrates from MAF2 to MAF3, which may result in some visible changes.

6.3. Particle Based Pathway

In this pathway, the muscles in their rest-pose positions are decomposed into fibres using any of the existing muscle decomposition methods, apart from *Updated Particle*. After that, these fibres are wrapped around the bones (this is the *Updated Particle* method) and updated surface models for the muscles are constructed from the wrapped fibres. Penetration between the muscles and the bones is automatically avoided in this pathway, and the process is completed in a reasonable time, even for the whole atlas. On the other hand, the volume of the muscles being wrapped may not be well preserved. Furthermore, the current method for the reconstruction of the surface from the wrapped fibres tends to produce artefacts at places where the muscle is bent.

To enable this pathway, select *Mass-spring system method* in the settings of the motion fused musculoskeletal model VME (Fig 6.4). There are also other important settings for this pathway. The *Enable constant time step* checkbox allows switching between constant and adaptive iteration time steps – using adaptive time steps decreases the probability of penetration between objects but has a trade-off of a greater computation time.

If constant time steps are used, control of the frequency is via the *Iteration time step* editbox – a larger time step leads to a faster processing but this increases the risk of penetration between objects. The *Iteration time step* value should be a positive integer; a maximum value is not specified, but it should be smaller than the *Maximum IterNum* value or the penetration avoidance mechanism is disabled.

The last available setting is *Maximum IterNum*, which specifies the maximum number of iterations that the Mass-spring system method may perform. Larger values will give a more precise wrapping but will necessitate larger computation times. We note that the method may terminate early if the system becomes stable before the *Maximum IterNum* number of iterations is reached.

As in the Energy Minimization pathway (Section 6.2), it is possible to specify bones or muscles that should be ignored during the wrapping process. To do so, the option *Enable filtering* should be checked; the models to be ignored are then specified in the list at the end of the settings.

> **CAUTION:** The rest-pose muscle fibres to be wrapped are obtained by running the decomposition method configured in the corresponding muscle wrapper in the rest-pose atlas. This requires the surface wrapping method to be run first (with the current-pose equal to the rest-pose), so it is important to pay attention to how the rest-pose musculoskeletal model VME is configured, for reasons described below.

As an example, we suppose that the rest-pose (static and already morphed) musculoskeletal model VME is *Generic Atlas* and the motion-fused musculoskeletal model VME is *Generic Atlas fused with Walking*. Both models contain a muscle wrapper with the name *Iliacus*, but for simplicity, let us rename the one under the rest-pose musculoskeletal model *Iliacus_RP* and the other *Iliacus_CP*.

If the motion-fused model is configured to use the Mass-spring system method, once the *Iliacus_CP* muscle wrapper VME is selected, and provided that *Iliacus_CP* is not included in the list of models to be ignored, the corresponding *Iliacus_RP* is located and its associated muscle is wrapped around the static bones using the pathway which is configured in its parental *Generic Atlas* VME.

It is important to point out that the Particle based pathway may not be used, so the surface of the muscle is first wrapped and then decomposed into muscle fibres. It is recommended that the Mesh skinning pathway is used as the drawbacks of this pathway do not manifest themselves when there is no movement, and this pathway is very fast.

As soon as the surface of *Iliacus_RP* is wrapped, it is decomposed using the decomposition method specified in this muscle wrapper (i.e., *Iliacus_RP*); the decomposition is performed even if this is not set. The muscle fibres obtained from *Iliacus_RP* are now wrapped around bones to accommodate in Iliacus_CP. After that, the surface of Iliacus muscle in the current-pose is reconstructed from the

wrapped fibres and, if any method other than the *Updated Particle* decomposition method is set for *Iliacus_CP*, this surface is further decomposed using the method selected.

It is expected that the above configuration will be simplified during the migration from MAF2 to MAF3.

6.4. Muscle Surface Wrapping Methods

In this section, we describe all methods for the wrapping of muscle surfaces that are currently available in LHPBuilder. The method to be used is specified in the settings of the musculoskeletal model VME (see the previous sections). We note that to visualize the wrapped muscle surface, one should uncheck the *Generate fibres* checkbox in the settings of the muscle wrapper VME (see Fig. 6.3).

For debugging purposes, the settings of muscle wrappers provide two special checkboxes – see Fig. 6.5. When *Show input* is enabled, the system displays a new window with all bones and muscles that participated in the wrapping (i.e., were not ignored – see Sections 6.2 and 6.3) in both their rest-pose and current-pose positions (i.e., before and after muscle surface wrapping). The rest-pose musculoskeletal model is green; the current-pose one is yellow. The wire model in the visualization is the rest-pose model after being rigidly transformed so that the rest-pose pelvis matches the current-pose pelvis. The *Show progress* option is valid only in the case of Energy Minimization pathway and displays the successive wrapping of muscle surfaces. To skip the visualization, press key *X*.

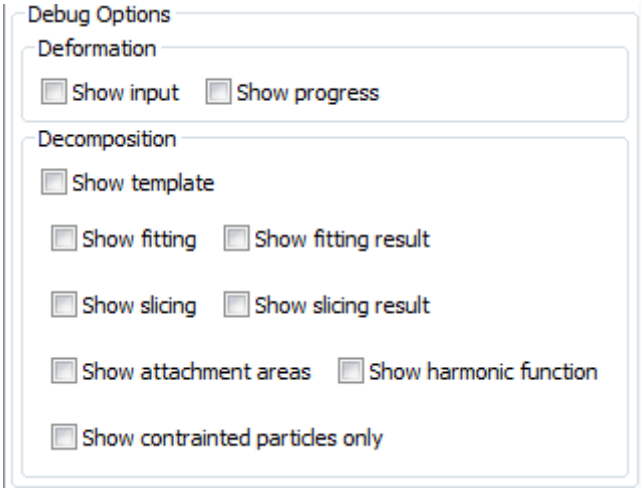


Fig. 6.5. Options of medVMEMusculeWrapper for muscle surface wrapping debugging

6.4.1. Fast Skinning (LHDL)

This method requires that each muscle is associated with one or more action lines describing its general path between the attachment areas on the bones. In contrast to most action-line models, however, only one action line has to be specified for a muscle in our model since an action-line loses its mechanical function and is considered only as the muscle "skeleton".

In our model, an action line is represented by a wrapper VME, which may be *mafVMEMeter* or *medVMEComputeWrapping*. While *mafVMEMeter* supports poly-lines going through predefined via points (landmarks), *medVMEComputeWrapping* is capable of calculating the path of an action-line that wraps around a predefined set of obstacles. In fact, one wrapper VME represent an action line in either the rest-pose or the current-pose position, so two wrappers must be associated with the muscle in the settings of particular muscle wrapper VME – see Fig. 6.3. In the rest-pose (static) musculoskeletal atlas, both wrappers are the same, and when the atlas is being fused with the motion data, the wrapper for the current-pose is automatically updated to be time-variant.

If an action-line is a simple line segment (which is quite a common case), it is impossible to detect the rotation of its muscle around this line segment from the positions of the rest-pose and current-pose

wrappers, so one must also specify some reference system VME for each wrapper. It is recommended that a bone along which the muscle runs is used for this purpose.

For each time frame of the simulation (the current-pose position), the path of action line represented by the current-pose wrapper is recalculated, and the difference between this path and the previous path (described by the rest-pose wrapper) determines how the muscle surface should change to fit the anatomy at the current time frame. We exploit real-time techniques for the skeleton-based animation of surface models that are well known in computer graphics to deform the muscle accordingly, while trying to preserve the volume of the muscle being deformed – if the path of the action line has become shorter, the muscle should bulge in order for its volume to be preserved; if it wraps around an obstacle, the muscle must bend and stretch. In most cases, the error in the volume was found to be below 5%, on average, thought the maximal error experienced has exceeded 13%. Each muscle is processed independently of the others, leaving the responsibility of having an acceptable co-penetration between muscles to a proper definition of their action lines (skeletons). On the other hand, meshes with thousands of vertices can be deformed in milliseconds on commodity hardware by this technique.

SUMMARY: Used in the Mesh Skinning Pathway only; fast (15 ms per muscle); volume preservation errors below 5% on average; high risk of penetration between muscles and bones; recommended for the rest-pose musculoskeletal atlas when methods for muscle decomposition are tested.

6.4.2. PK Method

This method also demands that each muscle is associated with one or more action lines that describe its general path between the attachment areas on the bones. The paths of the action lines should be accurate – if not, unrealistic shapes of the wrapped muscles may be produced (typically, a muscle no longer touches the bone at one side), which is not found with the fast-skinning method.

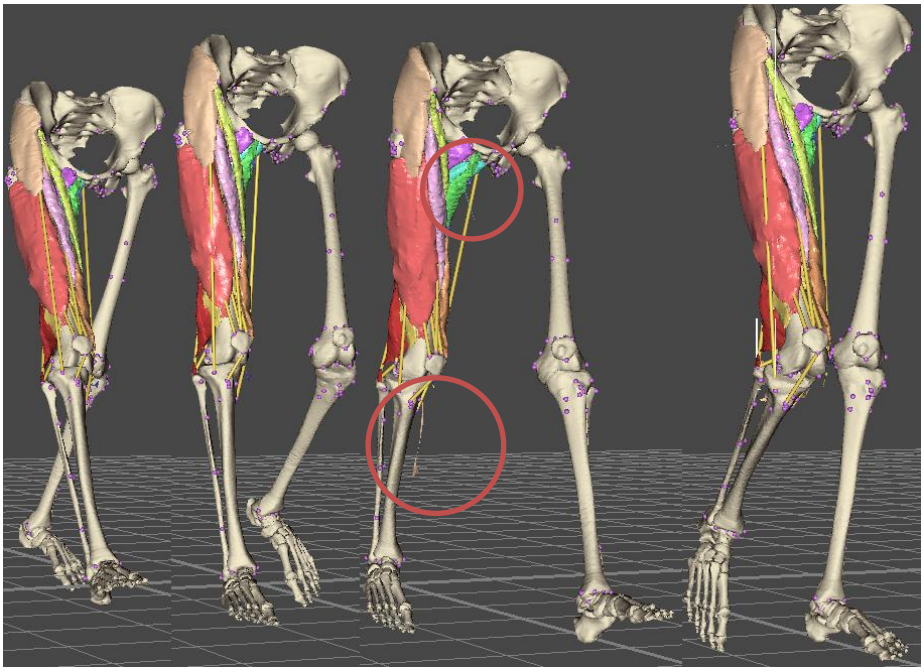


Fig. 6.6. Non-manifold muscles wrapped using the PK method, with evident artefacts.

The PK method exploits total energy minimisation. In our case, the energy of each vertex of the input muscle surface mesh is derived from its position relative to vertices in its local neighbourhood and from various "soft constraints" (e.g., the relative position of the vertex from action lines) or "hard constraints" (e.g., the participation of the vertex in relation to the total volume). If any constraint changes, the energy also changes, and the technique then tries to reposition the vertices of the original mesh in order to minimise the total deformation energy. We note that, for meshes that are non-manifold or

have boundaries or self-intersections, the computation of energy is unstable, so the results produced are unpredictable – see Fig. 6.6.

The method creates a system of non-linear equations that are solved by an iterative Gauss-Newton method with Lagrange multipliers; this is not only time and memory consuming, but also numerically unstable. Because of this, an initial solution (or predictor) is found using a coarse control mesh, which must preserve the rough shape of the original mesh and should fully contain it; this is then corrected by a few final iterations in which the full mesh is used. The coarse control mesh for the muscle surface is calculated automatically when the method encounters the muscle for the first time and stored (as a surface VME in the *Hull* folder) for later re-use, i.e., it is calculated in the first frame only. There are two algorithms available for the construction of this mesh; it is recommended that *Progressive Hulls* is used (see Fig. 6.4). Although time-consuming (it may take one minute per mesh), it increases the numerical robustness of the method.

The PK method may be run with penetration prevention turned off or on. In the former case, each muscle is processed independently, leaving the responsibility for having an acceptable co-penetration between muscles to a proper definition of their action lines (skeletons), as in the fast-skinning method (Section 6.4.1). In general, the processing of one muscle in this mode requires several seconds, and the maximal loss in volume is lower than 0.4%.

If penetration prevention is turned on, all muscles and bones not specified to be ignored (see the description of the energy minimization pathway in Section 6.2) are processed simultaneously and the method tries to avoid penetration between objects.

This is achieved at a great cost. First, the current implementation in LHPBuilder takes easily 20 minutes even for a couple of objects involved in the process because the algorithm used to detect penetration is computationally demanding. Next, as penetration prevention is given the highest priority, the method may not be able, at the same time, to preserve the volume (errors have been found to be comparable with the fast-skinning method – see Section 6.4.1). Finally, if some penetration was detected during the wrapping, the final wrapped muscle surface is typically no longer smooth and the volume taken off to prevent penetration is readily apparent, as can be seen in Fig. 6.7.

Note that when action lines are inaccurate, it may happen that a muscle penetrates the entire volume of a bone – an inaccuracy of the action line of a muscle may cause the muscle to be pushed into the bone, so a small penetration is detected and corrected, which in turn only increases the pushing force for the next iteration. When the pushing force becomes so large that part of the muscle jumps through the whole bone, the penetration prevention mechanism fails to detect the problem and no corrective action is taken.

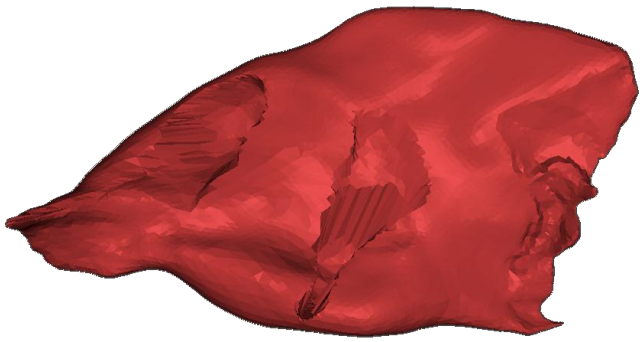


Fig. 6.7. Gluteus Medius after wrapping by the PK method with penetration prevention.

However, the biggest problem of the PK method running in this mode is that, in many cases, it does not work correctly because it takes muscles in their rest-pose, and tries to wrap them according to their action lines in both the rest-pose and current-pose positions but bones, as obstacles that may not be penetrated, are taken in their current-pose. Hence, after the initial rigid transformation of the rest-pose musculoskeletal model, which registers the rest-pose pelvis to the current-pose one, it may happen that a muscle may already penetrate a bone, often significantly – see Fig. 6.8.

SUMMARY: Used in the Energy Minimization Pathway only; it requires the muscles and action lines to be of good quality; it works in two modes with different results:

– with *Use Multiple Objects* off: interactive times (less than one second per muscle), volume preservation errors below 0.4% on average, high risk of penetration between muscles and bones;
– with *Use Multiple Objects* on: extremely slow (minutes to hours depending on the number of muscles considered), volume preservation errors below 5% on average, low risk of penetration between muscles but very high risk of penetration between muscles and bones;

Thus, it is recommended for use of penetration detection either with all bones ignored or with only the pelvic area enabled; in all other cases, the *Interpolated PK Method* (see Section 6.4.3) should be used.

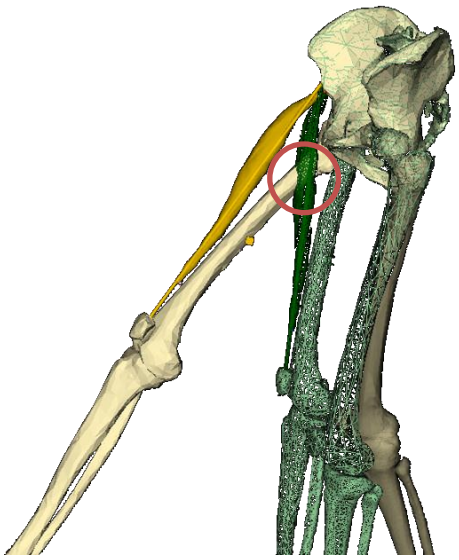


Fig. 6.8. Initial penetration of the Rectus Femoris and the Femur causing the PK method to fail.

6.4.3. Interpolated PK Method

This method supersedes the simple *PK Method* (Section 6.4.2) when avoidance of penetration between muscles and bones is the primary objective. It interpolates positions between the rest pose and the current pose, running the original method for each pair of neighbouring positions. Currently, three steps are used between the rest pose and the current pose, but further experimentation may allow this to be changed so that the number of steps is decided adaptively.

As an example, let us assume that we have three positions: the rest-pose (in the static atlas) = P_0, P_1, and P_2 = the required current-pose. The paths of the action lines and the positions of the bone vertices at P_1 are calculated by linear interpolation between those at P_0 and P_2. The *PK Method* is then run to wrap muscles from P_0 into P_1, which provides us with the starting positions of the muscle vertices for a further running of the *PK Method* to wrap them to P_2.

An advantage of this method is that it diminishes the problem of penetration between the bones and muscles reported in the previous section. Furthermore, places where some penetration was detected during the wrapping are less apparent – compare Fig 6.9 with Fig. 6.7. On the other hand, coarse control meshes (hulls) for muscles must be constructed before each run of the *PK Method*, which increases the overall time significantly. Provided that the muscles are of good quality, the *Use Progressive Hulls* option (see Fig. 6.4) may be turned off to speed up the process.

SUMMARY: Used in the Energy Minimization Pathway only; requires the muscles and action lines to be of good quality; works in two modes with different results:

– with *Use Multiple Objects* off: slow (minutes per muscle); volume preservation errors below 0.4% on average, high risk of penetration between muscles and bones;

– with *Use Multiple Objects* on: extremely slow (hours, depending on the number of muscles considered); volume preservation errors may exceed 5%; low risk of penetration between objects (muscles, bones)

it is recommended for use in comparisons with the Mass-spring system method (Section 6.4.4).

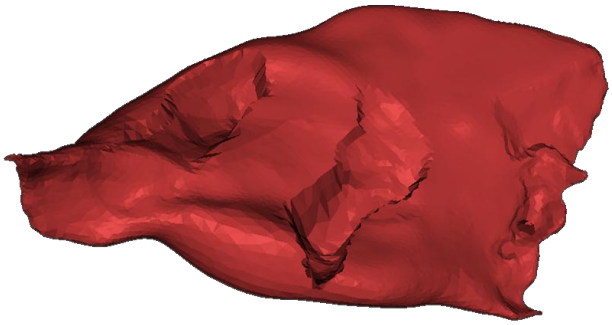


Fig. 6.9. Gluteus Medius after wrapping by the Interpolated PK method with penetration prevention.

6.4.4. Mass-spring system method

This method uses point-mass particles connected by fictional springs to represent the deformable object. The positions of the particles are obtained by sampling the rest-pose muscle fibres that stretch through the interior of the muscle; they are connected by springs according to the pattern selected in the settings of the current-pose muscle wrapper VME of this muscle – see the description in Section 6.5 and Fig. 6.12, later.

It is required that muscle fibres are sampled uniformly within the volume and that their number is equal to the square of an integer. The method automatically updates the settings of the decomposition method used to produce the fibres so that this requirement is always fulfilled, even if the original settings of the rest-pose muscle wrapper for this muscle are not fully compliant with it.

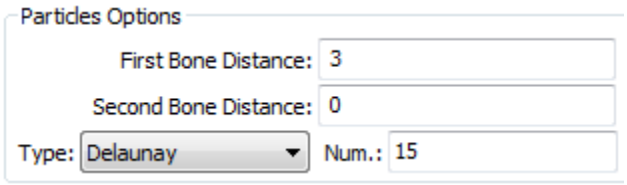


Fig. 6.10. Options of medVMEMuscleWrapper for particle construction.

Each particle on the boundary of the fibres is checked to find out if it lies in the proximity of a bone – the threshold distance can be configured in the settings of the current-pose muscle wrapper (see Fig. 6.10). If it does, it is fixed to its closest bone and moves with it thereafter, thus inducing a movement of the other particles in order to keep the entire mass-spring system in balance. The process is iterative and stops when the mass-spring system restores balance or the maximum number of iterations is reached.

The method automatically attempts to prevent penetration between particles from different objects (muscles or bones). When the process is finished, the surface model of the muscle is reconstructed based on the modified positions of the particles. Although places where penetration was prevented are not immediately apparent (they are smooth, unlike in both of the *PK* methods), the current algorithm for reconstruction tends to produce artefacts at places where the muscle is bent – see Fig. 6.11. Another algorithm for the surface reconstruction is currently under development.

Even though the method is currently implemented on the CPU only, it can still process the whole atlas in minutes (depending on the maximum number of iterations). We note that the first frame is always much slower than the others as the rest-pose muscle fibres must be computed.

SUMMARY: Used in the Particle Based pathway only; slow (minutes per whole atlas); volume preservation errors as yet not quantified, but may exceed 5% on average; low risk of penetration between objects (muscles, bones); recommended for use as the default option.

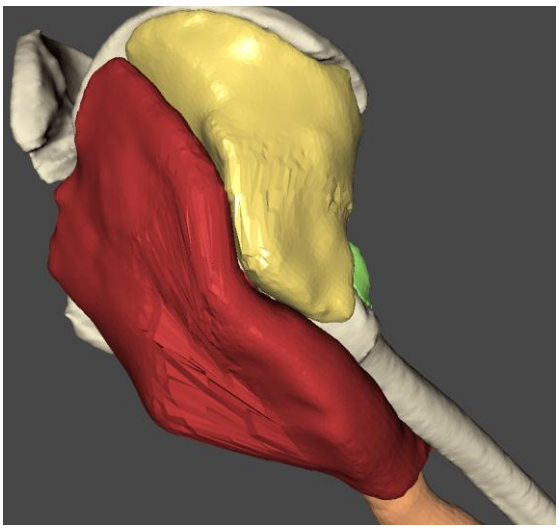


Fig. 6.11. The result for all three Gluteus muscles and Biceps Femoris.

6.5. Muscle Decomposition Methods

In this section, we describe all methods that are currently available in LHPBuilder for decomposing the muscle volume into muscle fibres. The method to be used is specified by the user in the settings of the muscle wrapper VME – see Fig. 6.12; to visualize the muscle fibres constructed, the *Generate fibres* checkbox should be checked. For each muscle, two landmark clouds, denoting the origin and insertion area, should be specified. The landmarks in the clouds should be oriented, i.e., they should form a closed contour that does not self-intersect, or unexpected behaviour may result. If the attachment area can be approximated by a single point, it is possible to omit this specification and the available methods will automatically access this point from the muscle action lines.

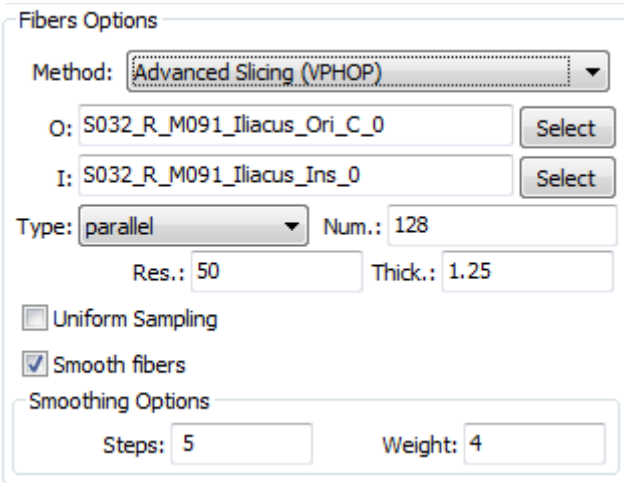


Fig. 6.12. Options of medVMEMusculeWrapper for muscle decomposition.

Other settings control the process as follows. *Type* selection allows the architecture of the fascicles within the muscle to be specified – *Parallel* provides good results in most cases. We note that this option is completely ignored by the *Mass-spring System* method.

Fibres are distributed within the muscle volume in either a uniform grid layout or in a quasi-random fashion depending on whether the *Uniform Sampling* option is checked or not; the *Mass-spring System* ignores this setting and always uses *Uniform Sampling*. *Uniform Sampling* is also used if the fibres produced are to be the input for the *Mass-spring* muscle surface wrapping method (see Section 6.4.4).

Num. specifies the number of fibres that are to be produced; however, a decomposition method may over-ride this value, if necessary. *Res.* is the resolution of a fibre, defined as the number of line segments from which the fibre will be formed. Naturally, a higher resolution leads to greater computational times, but the fibres produced will probably have a smoother, more natural, look. *Smoothing* is iterative: the more steps that are used, the smoother the fibres will be (see *Steps*). *Thick* specifies the thickness of a fibre and is used purely for visualization.

For debugging, the settings of the muscle wrappers also provide several special checkboxes – see Fig. 6.13. When checked, a window visualizing the current state is typically displayed. To skip the visualization, press key *X*.

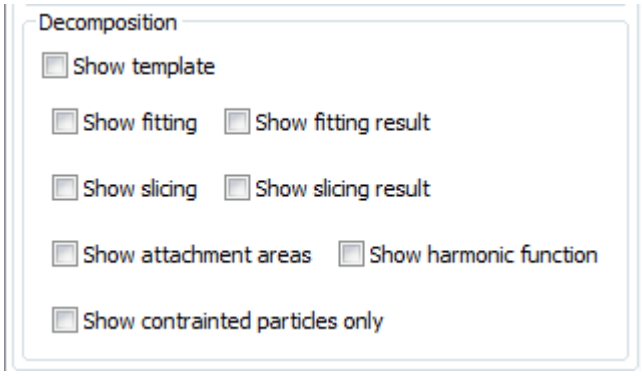


Fig. 6.13. Options of medVMEMusculeWrapper for muscle decomposition debugging.

6.5.1. Simple Slicing (LHDL)

This method starts with the production of poly-line muscle fibres of the requested number (*Num)* and resolution (*Res*) within a unit cube according to the fibre architecture [8] specified for the muscle being decomposed. Next, the unit cube (and all of its poly-lines) is subjected to an affine transformation such that the transformed cube is an oriented bounding box (OBB) of the muscle, and the attachment sites of the fibres in the cube are aligned as well as possible with those specified for the muscle (as landmarks). After this, the transformed cube is sliced and the contours that arise from the slicing are morphed on to the contours of the muscle obtained by the same slicing, employing the technique described by Ju et al.[10], to ensure consistency between the slices. This maps the fibre vertices into the interior of the muscle.

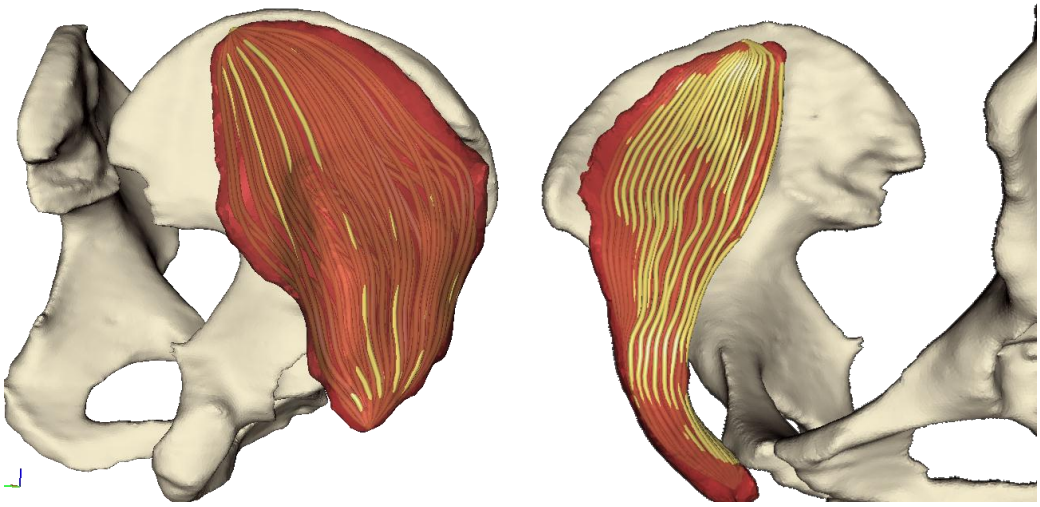


Fig. 6.14. Fibres of Gluteus Medius and Iliacus produced by the Simple Slicing method.

The method can decompose a muscle of the typical size (10,000 triangles) within one second even for very large number of fibres (256) at a high resolution (50 segments). As a resolution of 20-30 seg-

ments has proved sufficient for any muscle so far tested, and 50-100 fibres usually produce visually plausible results, the method is suitable for interactive visualisation. A drawback is that the method always produces fibres that meet at a common point (see Fig. 6.14), which is not the behaviour of fibres in all muscles.

SUMMARY: Fast (less than one second per one muscle); suitable only for muscles with attachment areas that can be approximated by a single point (e.g., Sartorius muscle)

6.5.2. Advanced Slicing

This method extends Simple Slicing by changing the path of the muscle fibres in the proximity of the attachment area to provide a better correspondence with reality. To do so, two cutting planes perpendicular to the principal axis passing through the extreme (in the direction of the principal axis) landmarks of attachment areas are constructed and used to cut out the unwanted parts of the fibres. In the following step, the cut parts of the fibres are reconstructed by extrapolation from the remaining (uncut) paths.

Although slightly slower than its simpler counterpart, the method still decomposes a muscle of a typical size within one second. As can be seen in Fig. 6.15, paths of most of the fibres produced correspond to these depicted in anatomical atlases. The Advanced slicing method, however, cannot guarantee a uniform distribution of the fibres over an attachment area; further, if an attachment area is nearly perpendicular to the cutting plane, the muscle fibres may clustered on one side of this attachment area.

SUMMARY: fast (less than one second per one muscle); especially suitable for muscles with attachment areas whose fitting planes have similar normal vectors.

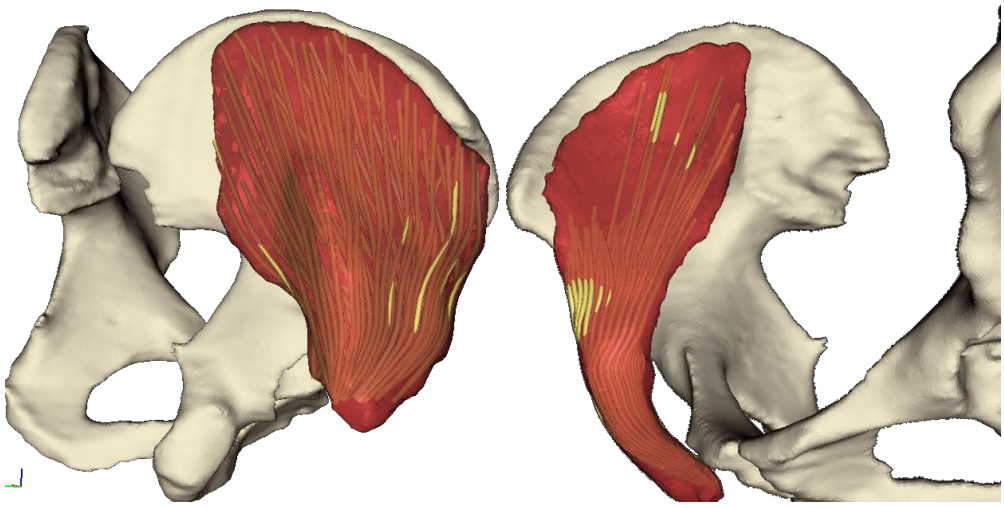


Fig. 6.15. Fibres of Gluteus Medius and Iliacus produced by the Advanced Slicing method.

6.5.3. Advanced Kukacka

Unlike the previous two methods, which slice both the template and the muscle using the same plane, this method slices the muscle according to the iso-value of a harmonic scalar function on its surface, which means that the muscle contour may be twisted, rather than planar. As a result of this, the paths of the muscle fibres in the proximity of the attachment areas correspond to what is expected without the necessity to somehow correct them (as in Advanced Slicing method), and, furthermore, fibres are uniformly distributed over an attachment area. On the other hand, the calculation of the harmonic scalar function requires several seconds. At present, this function is calculated each time it is needed, but we note that it could be calculated just once and stored as a scalar field with the muscle geometry.

SUMMARY: convenient time (about 5 seconds per muscle); suitable for any muscle, though fibres may not be smooth (especially, if any type other than *Parallel* is selected).

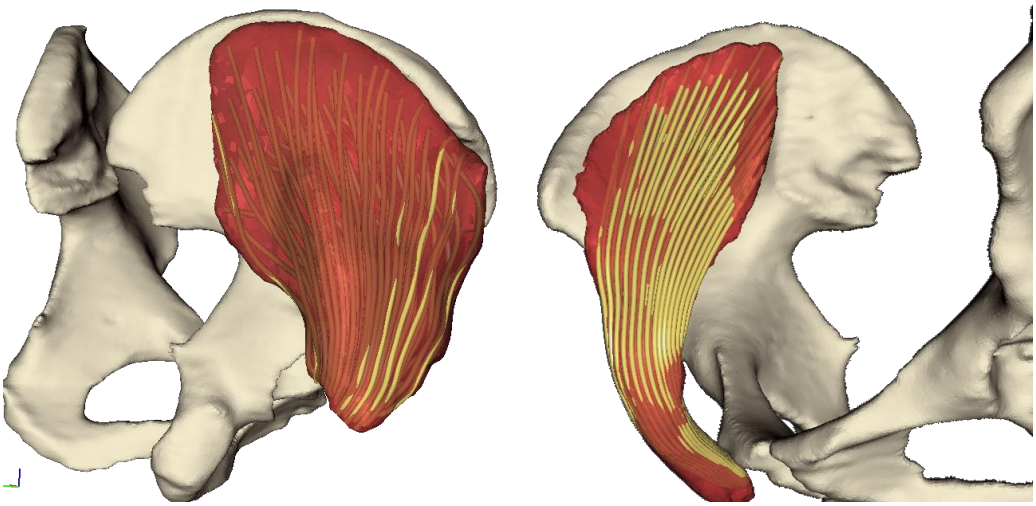


Fig. 6.16. Fibres of Gluteus Medius and Iliacus produced by the Advanced Kukacka method.

6.5.4. Mass-Spring System

This method starts by constructing a mass-spring system in which each muscle fibre is represented by a linear set of mass particles connected by springs. Adjacent particles from different fibres are also connected by springs so, in effect, the particles lie on a regular 3D grid. A *Parallel fibres* architecture is assumed (and used, even if not specified in the settings of muscle wrapper VME). Next, the medial axis of the muscle is computed, and the muscle is sliced by planes perpendicular to this axis to create a set of muscle contours; unlike the principal axis, which is a line, the medial axis is a curve.

The muscle contours are sampled uniformly to select the points on the surface to which the particles on the boundary are moved and fixed. This adjustment induces a movement of internal particles in order to keep the mass-spring system in balance; the process is iterative and stops when the mass-spring system restores balance.

As can be seen in *Fig. 6.17* and Fig. 6.18, the method produces fibres that are uniformly distributed within the volume and do not twist unnaturally, as may occur in the previous methods. The method is slow, taking easily one minute per muscle, although the current implementation runs only on the CPU.

SUMMARY: currently slow (about one minute per muscle); suitable for any muscle.

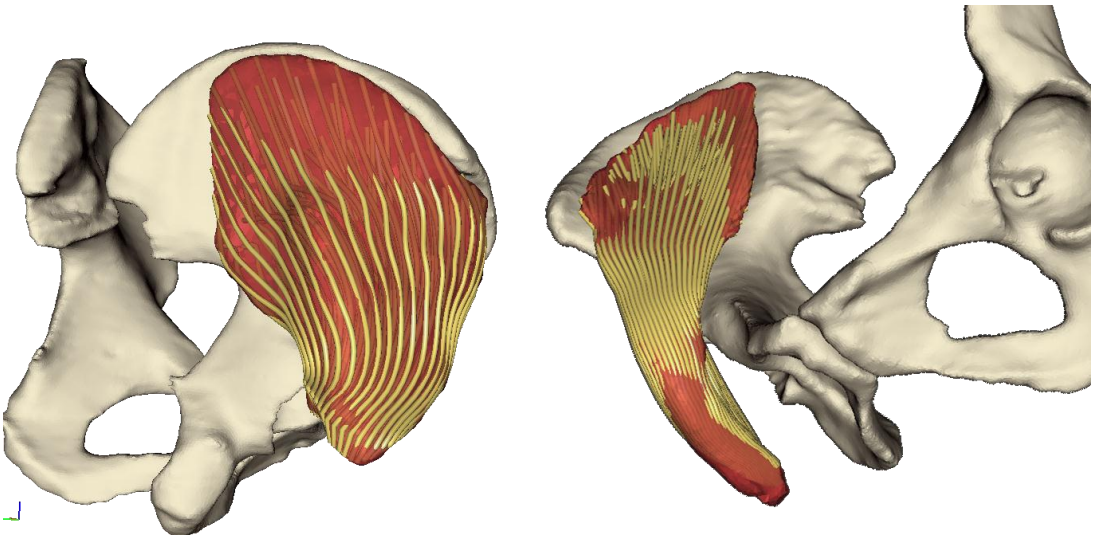


Fig. 6.17. Fibres of Gluteus Medius and Iliacus produced by the Mass-Spring System method.

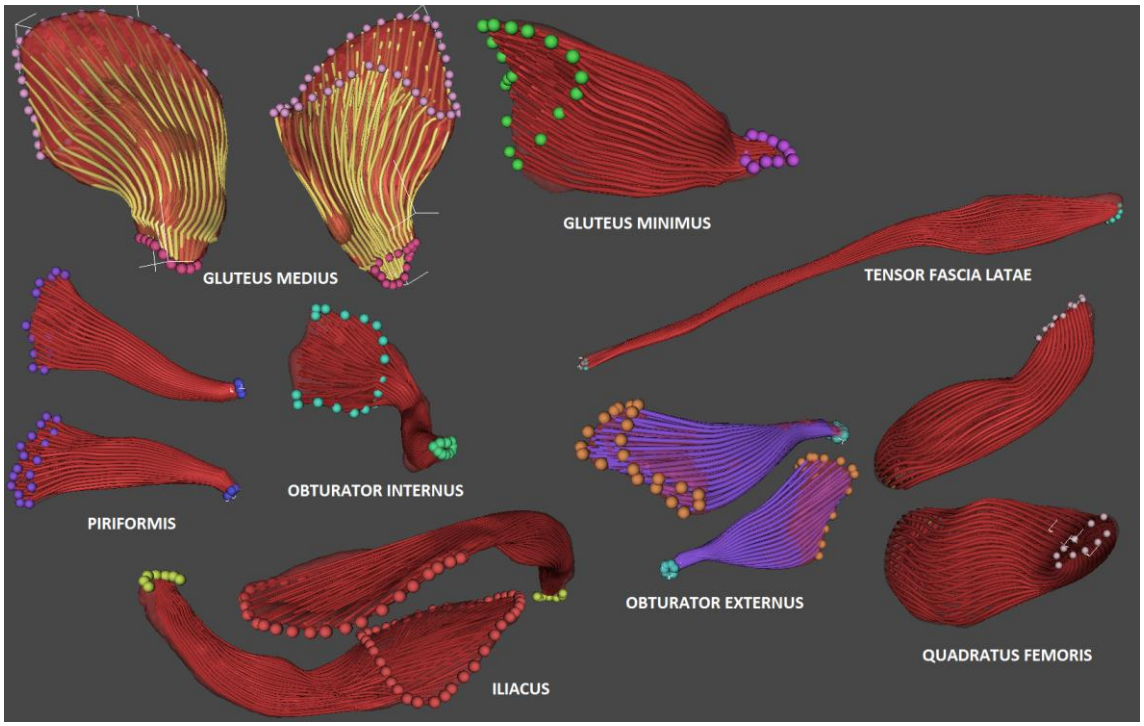


Fig. 6.18. Fibres of various muscles produced by the Mass-Spring System method.

6.5.5. Updated Particle

This method is completely different from the methods described above in that it does not perform decomposition of a muscle but simply provides the user with the result of muscle fibres wrapped using the *Particle-Based Model* pathway. The method is available for a current-pose muscle wrapper only and its result is dependent on the decomposition method (one of the previous methods) chosen for the corresponding rest-pose muscle wrapper. A sample result is presented in *Fig. 6.19.*

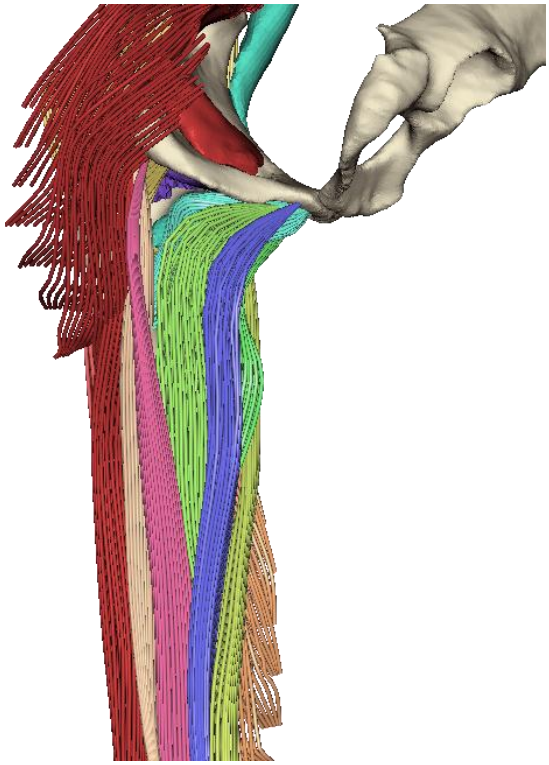


Fig. 6.19. Fibres of various muscles produced by the Updated Particle method.

Particles representing a muscle are visualized as a landmark cloud by the *Create [WP10]/Create Particles* operation run on the current-pose muscle wrapper VME for that muscle. This operation may take more than 20 minutes to complete because of inefficiency of the landmark cloud VME (developed by a third party during the course of other projects).

SUMMARY: suitable for any muscle in the current-pose; cannot be used in the rest-pose; valid only in combination with the *Particle Based Model* pathway.

6.5.6. Cadaver Fibres

This is another special method for muscle decomposition. It requires that several muscle fibres, represented by poly-lines, on the surface of a muscle that is to be decomposed are available, as well as information about the tendons. Usually, these can be provided when the same subject whose medical images were used to create generic musculoskeletal model is autopsied and muscle fibres are marked by optical markers and their digital models then created from these markers. In our musculoskeletal *Generic Atlas* model, the required data is stored in a group under the *Muscle Geometry* group.

The *Create [WP10]/Decompose Muscle* operation is run on the muscle to be decomposed. The user has to select a set of fibre data that corresponds to this muscle (as the filter cannot determine the semantics of the data). This can be done by simply selecting the proper dataset in the popup dialog after the operation is executed (see Fig. 6.20). We note that not every muscle available in our atlas has a set of this data, and many muscles may have the data available only for one of the limbs (for example, the data for the right-side muscle *S032_R_Iliacus BES* is available, but that for the left-side muscle *S032_L_Iliacus BES* is not). Be sure to select the matching data for the filter to work properly.

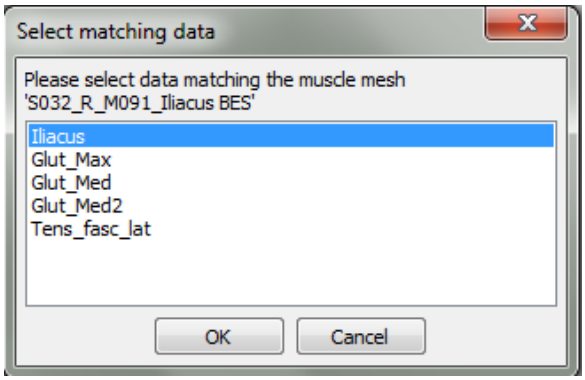


Fig. 6.20. Muscle decomposition - Matching data selection dialog.

When the matching data is selected, the main operation GUI opens, as shown in Fig. 6.21. The user is required to classify each input poly-line as a fibre or a tendon. This can be done by pressing buttons (a) and moving the selected items from the *Unassigned* list to the correct one: *Tendons* or *Fibers*, respectively. By selecting an item in any list, the appropriate data polyline is highlighted in yellow in the preview viewport to make the classification easier; Muscle fibres are highlighted in red, tendons are highlighted in black. For some muscles, the order of the fibres in the data set is inconsistent, which produces unexpected "inside out" results. To correct this, the fibres and tendons can be reordered using buttons (b). We note that spatial ordering of the fibres should be introduced in order for the filter to work correctly.

The description of the settings (c) is as follows. *Fiber subdivision* determines how smooth the resulting fibres will be (i.e., the number of resulting line segments per source line segment). *Surface subdivision* determines how many fibres will be created between two neighbouring fibres, and how many fibres will be generated inside the muscle itself. The *Show mesh* checkbox determines if the surface muscle mesh is displayed to the user; the *Show data* checkbox determines if the fibre/tendon data is displayed; and the *Show Result* checkbox determines if the muscle fibres produced should be displayed after pressing the *Preview* button.

An example of a successful decomposition can be seen in *Fig. 6.22*. The fibres in the *Preview* viewport are thick to improve their visibility. After pressing the *OK* button, the poly-line representation is stored as a new VME into the data tree.

SUMMARY: suitable for muscles in the rest-pose for which specific autopsy data is available; cannot be used in the current-pose; suitable for verification of the paths of fibres produced by other, artificial, decomposition methods.

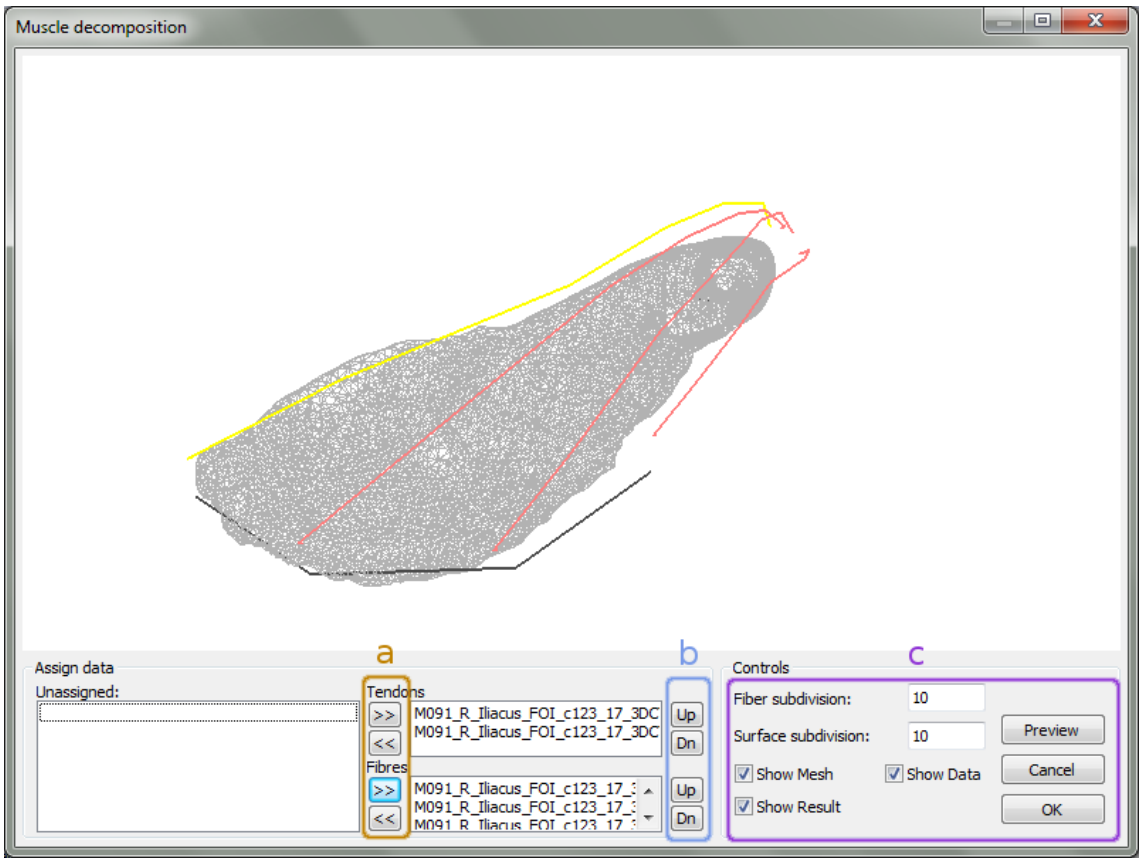


Fig. 6.21. GUI for the Decompose Muscle operation

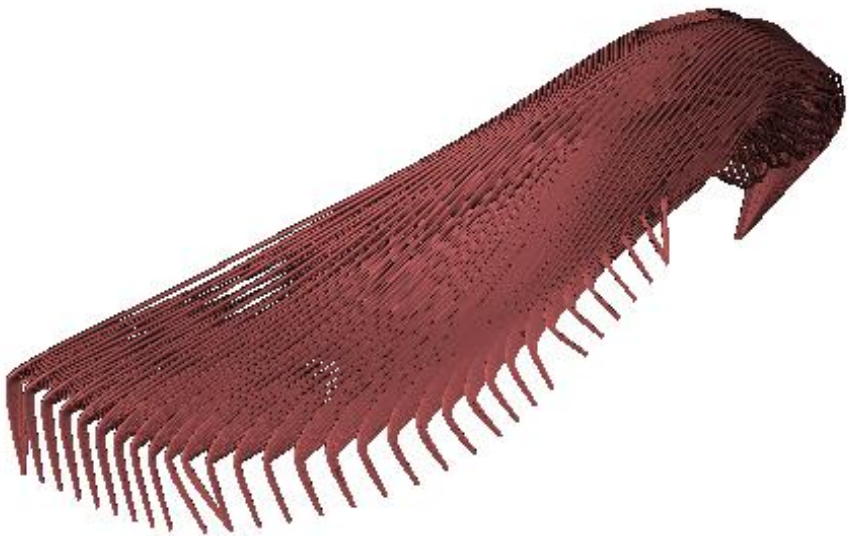


Fig. 6.22. Iliacus muscle decomposed using the Decompose Muscle operation

6.6. Progressive Hulls

Methods to wrap a muscle surface mesh that are in the *Energy Minimization* pathway work with a coarse outer hull of the mesh. The hull is constructed automatically and stored as a new surface VME in the *Hulls* group present in the rest-pose musculoskeletal atlas. An example of the hull can be seen in Fig. *6.23*.

It may happen that the automatically constructed hull is too coarse (this is particularly true for complex objects); this will causes problems when penetration prevention is turned on. In such cases, the hull can be created manually using the *Create [WP10]/Compute Hull* operation – see *Fig. 6.24*. The *Decimation* value determines the detail of the coarse mesh produced, whilst the *Quality* parameter influences the shape of triangles in the hull. The operation works in both CPU and GPU modes but the GPU version is less stable, so the CPU version is recommended for use in all cases.

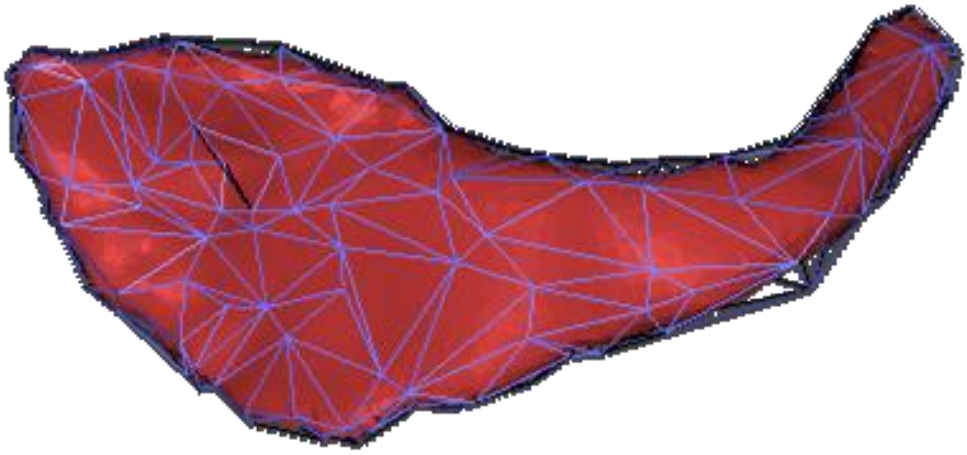


Fig. 6.23. *Iliacus muscle and its automatically constructed outer hull.*

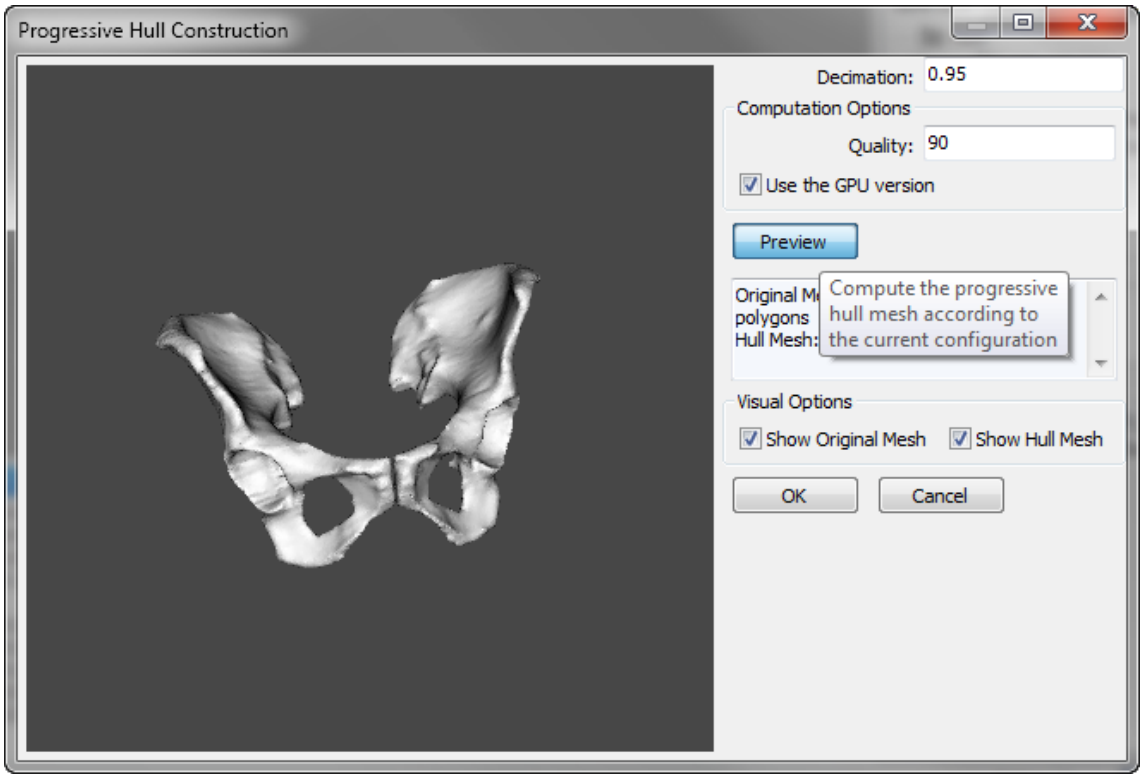


Fig. 6.24. GUI for the Compute Hull operation

7. Extending the Generic Atlas Model

If you decide to extend the generic atlas model (or even to create your own), there are a few things you should be aware of.

First of all, it is vital that the surface representations of regions, bones and muscles are of a high quality, i.e., they must be closed smooth manifold triangular meshes without self-intersections and without too irregular a structure (i.e., the sizes of triangles should not vary significantly; long, narrow triangles should be avoided, etc.). Failure to ensure this typically results in artefacts of various kinds. In LHPBuilder, there are various operations (to be located in the submenu *Filtering [WP10]* – see Fig. 7.1) that may help to improve the quality of your surfaces.

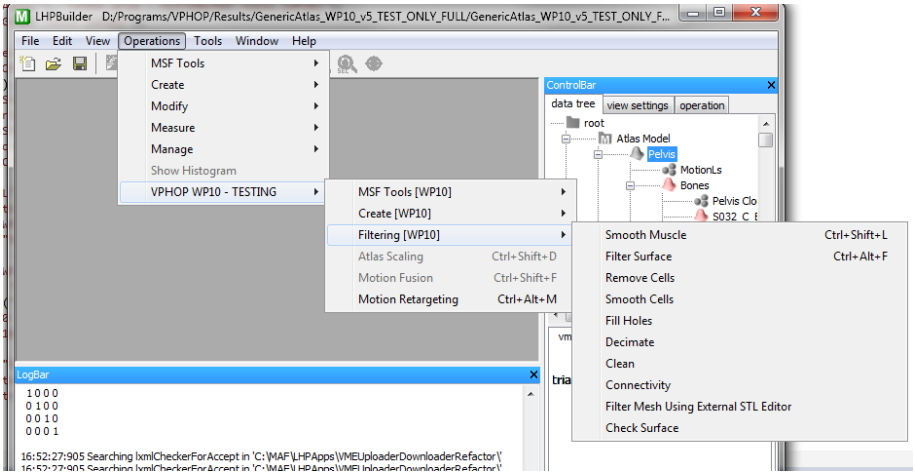


Fig. 7.1. Filtering options in the VPHOP WP10 menu.

If a mesh is too large (tens or hundreds of thousands of triangles), the processing may take a long time, or even crash because of insufficient memory for the internal structures. Hence, it is recommended that a lower resolution of the mesh is created and stored in another VME; in fact, one can create as many resolutions as required. However, LHPBuilder does need to be informed about the lower resolutions – this is not done automatically, even if you use decimate operation present in the application.

To do so, run *MSF Tools [WP10]/Set Lower Resolution…* on the VME with the higher resolution and select in this operation the appropriate lower resolution VME – see *Fig. 7.2*. This operation also allows the specification of outer hulls (create these using the *Create [WP10]/Compute Hull* operation), so you can define your own hulls instead of having them computed automatically by the muscle wrapping method – see Section 6.

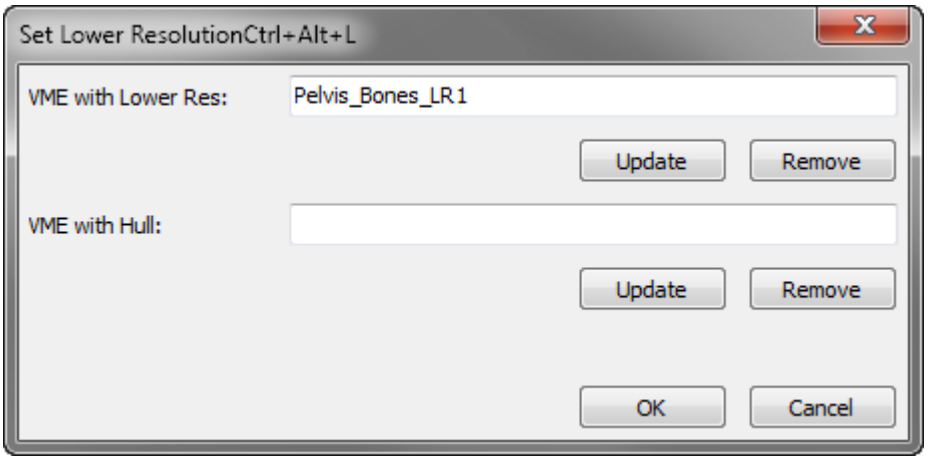


Fig. 7.2. Set Lower Resolution operation.

Procedural VMEs, such as wrappers, joints, muscle wrappers or even the musculoskeletal model root can be created by operations from the submenu *Create [WP10]* (see Fig. 7.1) and configured in their GUIs.

Please note that all regions, bones, muscles, joints and motion landmarks VMEs may have any name, however, they must have LHDL FA ontology tags assigned to them. These tags can be set using the *MSF Tools [WP10]/Edit Tag VME…* operation. If one object, e.g., the femur, has various levels of detail, all must have the tag assigned.

8. References

[1] Erdemir A, McLean S, Herzog W, van den Bogert AJ, Model-based estimation of muscle forces exerted during movements, Clinical Biomechanics 22 2 (2007) 131-154.

[2] Garner BA, Pandy MG, The obstacle-set method for representing muscle paths in musculoskeletal models, Computer Methods in Biomechanics and Biomedical Engineering 3 1 (2000) 1-30

[3] Jensen RH, Davy DT, An investigation of muscle lines of action about the hip: A centroid line approach vs. the straight line approach, Journal of Biomechanics 8 2 (1975) 103-110

[4] Delp SL, Loan JP. A computational framework for simulation and analysis of human and animal movement. IEEE Computing in Science and Engineering 2000, 2(5):46-55

[5] Arnold AS, Salinas S, Asakawa DJ, Delp SL, Accuracy of muscle moment arms estimated from MRI-based musculoskeletal models of the lower extremity, Computer Aided Surgery 5 (2000) 108-119.

[6] Audenaert A, Audenaert E, Global optimization method for combined spherical-cylindrical wrapping in musculoskeletal upper limb modelling, Computer Methods and Programs in Biomedicine 921 (2008)8-19

[7] Ng-Thow-Hing V, Anatomically based models for physical and geometric reconstruction of animals, PhD Thesis, University of Toronto, Department of Computer Science, 2001

[8] Blemker SS, Delp SL, Three-dimensional representation of complex muscle architectures and geometries, Annals of Biomedical Engineering 33 5 (2005) 661-673

[9] Ferrari A, Technical innovations for the diagnosis and the rehabilitation of motor and perceptive impairments of the child with Cerebral Palsy, PhD thesis, University of Bologna, 2010.

[10] Ju T, Schaefer S, Warren J, Mean value coordinates for closed triangular meshes, ACM Transactions on Graphics 2005, 243:561-566.