Contents

Part I

Introduction

Part II

Digital human

1 Generation of Meshes

The formulations of electrophysiology and contraction of tendon and muscle tissue are given on different domains. The muscle belly forms the domain Ω_M. A layer of fat and skin tissue on top of the muscle is denoted the body domain Ω_B. On both longitudinal ends, the muscle is attached to tendons, given by the domains $\Omega_{T,1}$ and $\Omega_{T,2}$.

All these domains are placed in the 3D Euclidean space, $\Omega_M, \Omega_B, \Omega_{T,i} \subset \mathbb{R}^3$. Additionally, formulation of electrophysiology on individual muscle fibers need one-dimensional domains $\Omega_{F,i} \subset \mathbb{R}^3$ for $i \in \{0, \dots, n_f\}$. In such formulation, the whole muscle typically consists of a large number n_f of these fiber domains. Each domain $\Omega_{F,i}$ is a 1D manifold embedded in the 3D domain. The domains are visualized in Fig. 1.1.

For discretization using the Finite Element Method, we approximate the 3D domains and 1D manifolds, Ω_{3D} and Ω_{1D}, by a number of 3D elements $\{U_{3D,i}\}_{i=1,\dots,n}$ with $U_{3D,i} \subset \mathbb{R}^3$, respective 1D elements $\{U_{1D,i}\}_{i=1,\dots,n}$, with $U_{1D,i} \subset \mathbb{R}^3$. An element is given by a number of nodes and their connectivity information. Their disjoint union approximates the overall computational domain, $\dot{\bigcup}_{i=1}^{n} U_{3D} \approx \Omega_{3D}$ and $\dot{\bigcup}_{i=1}^{n} U_{1D} \approx \Omega_{1D}$.

The elements can be defined by a mesh. In this section, structured meshes are assumed. A 3D structured mesh is homomorphic to a 3D regular cartesian grid. The number n of elements is the product of the numbers n_i, n_j and n_k of elements in the three coordinate directions x, y and z of the regular cartesian grid, i.e., $n = n_i\, n_j\, n_k$. Each element can be indexed by a triple (i, j, k) of indices with the ranges $i \in \{0, \dots, n_i - 1\}$, $j \in \{0, \dots, n_j - 1\}$ and $k \in \{0, \dots, n_k - 1\}$. In the program, typically, consecutive indices ι are used that

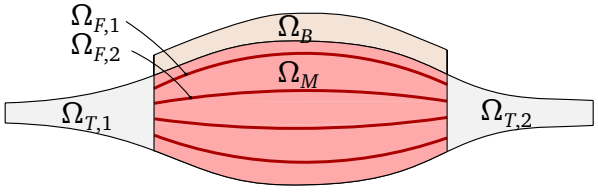


Figure 1.1: Visualization of the computational domains: tendons $\Omega_{T,1}, \Omega_{T,2}$, muscle belly Ω_M, body domain Ω_B and fiber domains $\Omega_{F,i}$.

iterate over all elements $\iota \in \{0, \dots, n-1\}$ and are obtained from the index triples by the mapping $(i, j, k) \mapsto \iota = k\, n_i\, n_j + j\, n_i + i$.

The elements can have different numbers of nodes, depending on the spatial order of consistency of the discretization. In this section, only 1D elements with two nodes and quadrilaterial 3D elements with $2^3 = 8$ nodes are considered. Higher order elements can be composed geometrically by using the nodes of the respective number of adjacent elements in the structured mesh. For this purpose, we always generate 3D meshes with an uneven number of elements in the coordinate directions. Thus, linear elements with eight nodes or quadratic elements with 27 nodes can be created from the same global set of nodes.

In the following, algorithms for the generation of the needed meshes are presented. The requirement is to construct a 3D structured mesh that fills a given volume. This is needed for the muscle and tendon domains, Ω_M and $\Omega_{T,i}$. Additionally, for the 3D muscle domain, a specified number of 1D muscle fiber meshes that are embedded in the 3D mesh have to be created. The fibers should be positioned to match the anatomy of the muscle. For the biceps and triceps muscles with their fusiform layout, a streamlined orientation of the fibers is needed.

1.1 Related Works

1.2 Preprocessing of the Muscle Geometry

The first step towards creating a structured mesh is to obtain a representation of the surface of the muscle. Starting point is a human biomedical imaging data set. In this section, two possible workflows are presented how to extract the muscle and tendon surface. The two workflows are visualized in Fig. 1.2. The workflow using the branch on the left side in Fig. 1.2 is automized but only works for the particular data set and extracting the biceps muscle. The right branch involves manual steps and is applicable for any muscle geometry.

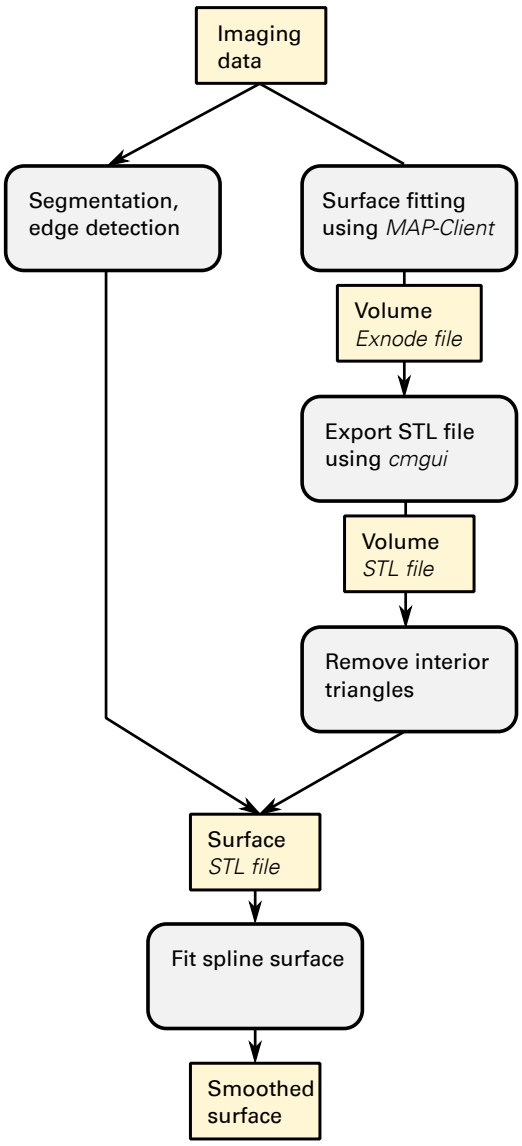


Figure 1.2: Workflow of generating a surface representation of the muscle and tendons from imaging data. Intermediate results are visualized as yellow boxes. Two possible branches are shown. On the left, the imaging data is processed using automatic processing to directly retrieve points on the surface of the muscle. On the right, the same is achieved with three steps of which the first one involves manual adjustments. At the end, a spline surface smooths the collected data from both possibilities to yield the resulting surface representation.

1.2.1 Data source

Anatomic images provide the basis for the extraction of muscle geometries. The used data set originates from the Visible Human Project [Spi96] of the United States National Library of Medicine. The project has published anatomic images derived from a male cadaver, among other data sets. The data, known as “Visible Human Male”, were published in 1994. Colored images of transversal cross sections were obtain by cryosectioning. A total of 1871 images with dimensions of 2048 by 1216 pixels and 24 bit color depth visualize the whole human body. Parts of the upper arms are contained in approximately 500 of these images. The size of a pixel is 0.33 mm in transversal direction and 1 mm in axial direction. The size of the complete set of JPEG compressed images is 772 MB. Cropping and selecting the relevant portions of the upper arm extracts a dataset with the size of 35 MB.

An extract of an image of the upper arm is given in Fig. 1.3. Biceps and triceps brachii muscles can be identified as the dark red tissue. For the biceps, the two muscle heads are visible, separated by the bright diagonal line from bottom left to top right. For the triceps, at least two of the three heads can be identified. The blue background is colored frozen gelatin that was needed during cryosection to stabilize the arms.

1.2.2 Automatic surface extraction

This section outlines the automatic algorithm to obtain the muscle surface from the Visible Human Male data set. The scheme corresponds to the left branch in Fig. 1.2. The algorithm was implemented as Python script as part of the Bachelor thesis of Kusterer [Kus19] that was supervised by me. The algorithm is capable of extracting muscle and bone geometries from the described imaging data.

At first, the color values of the images are used to segment the imaging data into muscle tissue, surrounding tissue and skeletal structure. The algorithm traverses the selected and cropped relevant parts of the images. For example, to segment the biceps muscle, the image region of pixels coordinates (x, y) with $x \in [1300, 1720]$ and $y \in [1030, 1720]$ are considered in the images with numbers 284 to 778.

For every such part of an image, pixels that match a certain range in the RGB color space are marked and categorized. The categories are muscle tissue and, for comparison, also bone tissue. The corresponding color ranges are given in Tab. 1.1. This color based classification does not succeed everywhere as the white shade corresponds not only to

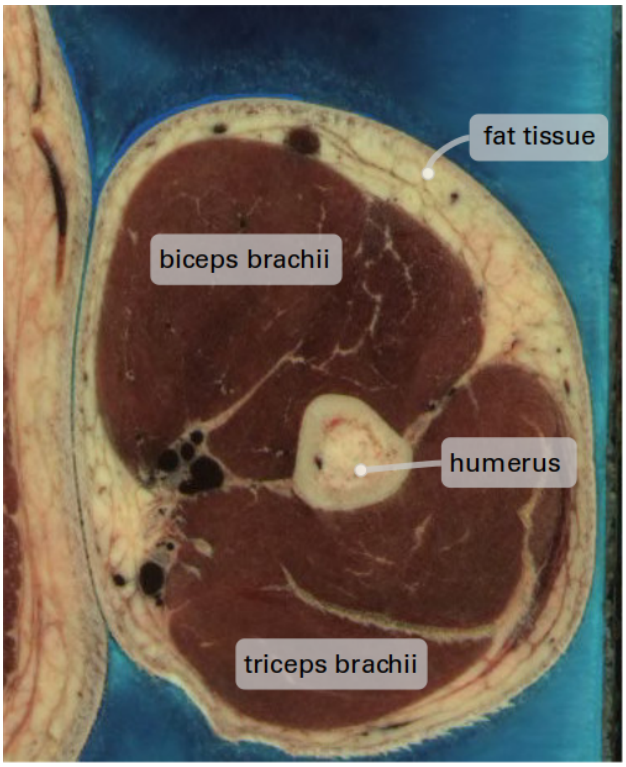


Figure 1.3: Exemplary extract of image number 483 from the Visible Human Male. A transversal slice of the left upper arm is shown as seen from the bottom. The biceps and triceps muscles as well as the humerus bone can be identified.

	red	green	blue
muscle	$60-100$	$30-75$	$15-60$
bone	$145-255$	$135-205$	$60-160$

Table 1.1: Ranges in the RGB color space to identify pixels of muscle and bone segments. The numbers correspond to 24 bit colors with the range $[0, 255]$ for every color channel.

bone material but also to fat and other tissue. Therefore, the algorithm removes artifacts located near the outer gelatine from the set of pixels categorized as bone.

Exemplary results for image number 483 are given in the left column of Fig. 1.3. It can be seen that the marked regions for muscle and bone have gaps in the interior resulting from differently colored tissue inside muscles and bones. On some images, the set of pixels also includes small objects outside the actual muscle and bone regions.

To reduce the gaps and small objects, the morphological operations *closing* and *opening* are performed on the data. These operations consist of *dilation* and *erosion* steps. Both are pixel based operations that traverse the dataset and for every pixel consider a window

of 3×3 pixels centered at the current position. Dilation picks the maximum value and erosion the minimum value from this window and assigns it as the pixel's value in a new image. In our case, values of zero and one correspond to non-categorized and categorized pixels, respectively.

Closing consists of dilation followed by erosion and closes small gaps or holes in the marked objects. Opening consists of erosion followed by dilation and removes small artifacts outside the actual bone and muscle areas. It was found effective to perform each dilation and erosion twice in sequence to yield good results with almost no more holes and unwanted small objects.

Next, the algorithm determines the contours of all regions with marked pixels. This leads to lines with a width of one pixel that enclose the muscle and bone areas. The right column of Fig. 1.4 shows the results after this step. It can be seen that the morphological operations close numerous gaps. In some images, as in the considered example, the muscle area gets split into multiple smaller enclosed regions which is not desired. However, proper contours of the biceps are found in the majority of images.

In the next step, a single contour for each of muscle and bone is obtained in every image. If there are multiple contours per image, the one that is located the most in upper right location within the image is selected for the muscle. If all contours in an image are shorter than 20 pixels, this is a sign of bad segmentation quality and the whole image gets discarded.

The result is a set of contours for muscle and bone in the cross-sectional planes of the images. Combining these, we get a point cloud in 3D space that approximates the surface of the biceps muscle and the surfaces of the considered bones humerus, ulna and radius. Using these points, a spline surface can be fitted and subsequently triangulated. Resulting surfaces for the biceps and humerus bones are shown in Fig. 1.5.

The runtime for the algorithm is 121 min on a AMD Ryzen 5 1600 processor with 6 cores, 3.2 GHz and 16 GB RAM, of which a maximum of 2 GB was used at maximum. Because processing of the images can be done in parallel, the runtime can be reduced to approximately half (62 min) using 2 threads and to a quarter (30 min) using 6 threads.

The advantage of the presented algorithm is that the outcome solely depends on the imaging data and, thus, no modeling error by manual approximation of the geometry occurs. For example, the obtained surfaces of biceps and humerus geometrically fit perfectly into each other. Intermediate steps are stored as black and white images. By

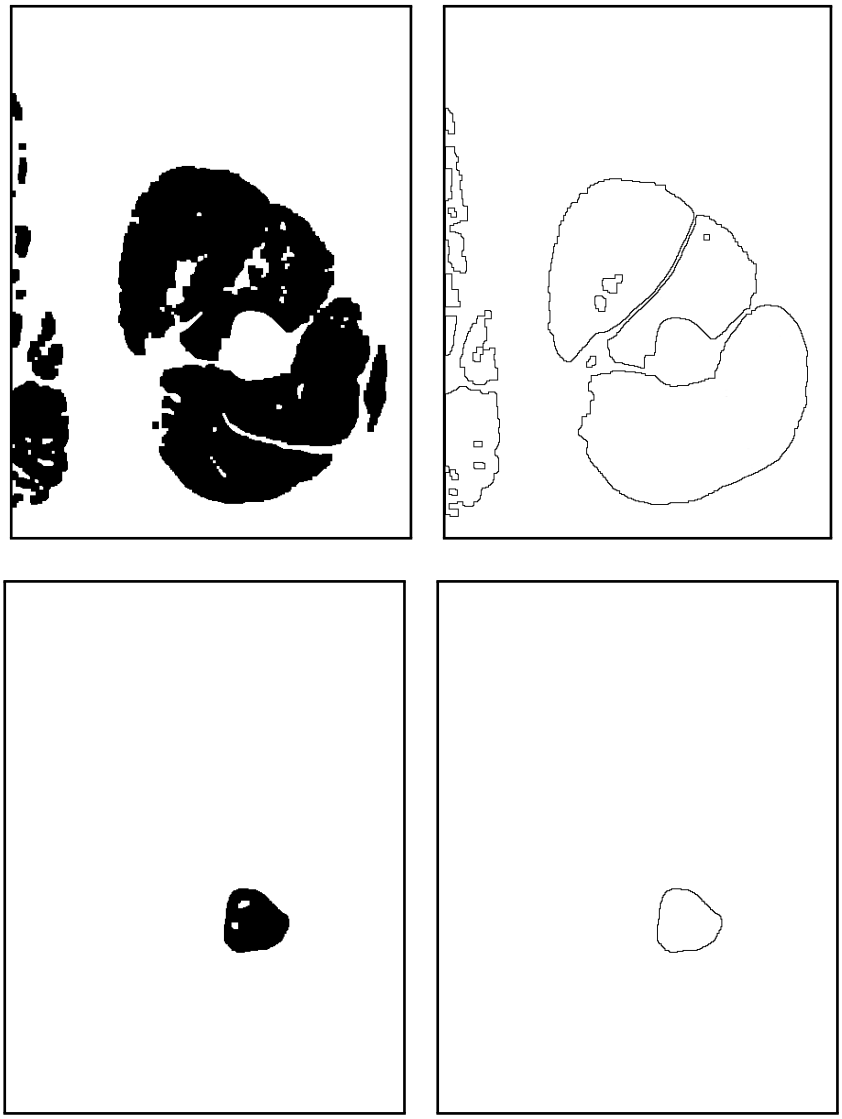
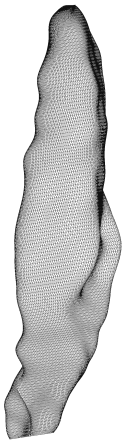


Figure 1.4: Intermediate steps of the algorithm to determine surface geometry of muscles and bones. The left columns shows pixels from the image in Fig. 1.3 that were categorized to be muscle tissue (top) and bone material (bottom). The right column shows a later step in the algorithm, where the surface of muscle (top) and bone (bottom) is estimated.



(a) Surface of the biceps brachii muscle. At the right side of the muscle, the groove of the humerus bone can be seen.

(b) Surface of the humerus

Figure 1.5: Resulting surfaces of biceps and humerus bone obtained by the automatic surface extraction algorithm.

editing these between the steps of the algorithm, manual tweaking is possible and could lead to increased quality of the results.

A disadvantage is that it relies on color information in the imaging data to differentiate between muscle and other tissue. Because the involved tissue has similar colors, these differences are often small. Furthermore, the color ranges need to be determined experimentally. Therefore, the algorithm is not very robust with respect to image noise and needs adjustments when it should be used to extract other muscles. Expert knowledge about the location and shape of human muscles cannot be used easily to improve the results of the algorithm.

A different approach is to manually segment the imaging data and construct surfaces with the help of a tool. This approach is described in the following section.

1.2.3 Manually guided surface extraction

The manually guided segmentation is done using the *MAP client* of the Musculoskeletal Altas Project (MAP) [Zha14]. This application allows to create and execute a workflow to achieve data processing and simulation tasks. In a graphical window, the user can place and connect various workflow steps. When executing the workflow, each step shows a dialog where the required configuration can be entered or the operations can be performed on a visual presentation of the data at this workflow stage.

Possible workflow steps include source and sink operations such as reading image data and writing meshes. Imaging data such as the 2D images from the Visual Human Male can be visualized in a 3D representation. The user can place points in the 3D space and try to match borders of the muscles and structures to extract from the data. Further workflow steps allow to create meshes of predefined geometrical shapes, such as cubes and cylinders and merge them into a common mesh. These meshes can be fitted to point clouds of user defined points. This is done by a least squares approach minizing the distances between user created points and the mesh surface.

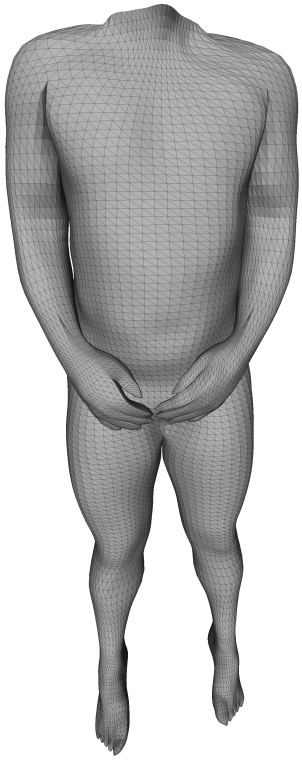
The MAP client has a plugin architecture and allows to create new workflow steps. It includes features from OpenCMISS, especially data processing formats and tools from OpenCMISS Zinc. Meshes can be created with 3D cubic Hermite elements that allow a high geometric modelling flexibility with a low number of nodes. Such meshes are stored in the OpenCMISS file format of `exnode` and `exelem` files.

As a result, meshes of individual muscles or the whole human organism can be created. Figure 1.6 shows meshes that were create from the cryosectioning data of the Visible Human Male. In Fig. 1.6a almost the whole body has been extracted. In Fig. 1.6b, the mesh consisting of cubic Hermite elements is visualized. A relatively coarse mesh width suffices to model a smooth surface of the body. When exported in the exfiles format from the MAP client, the data can be visualized, e.g., using *cmgui*, the visualization tool of OpenCMISS Zinc.

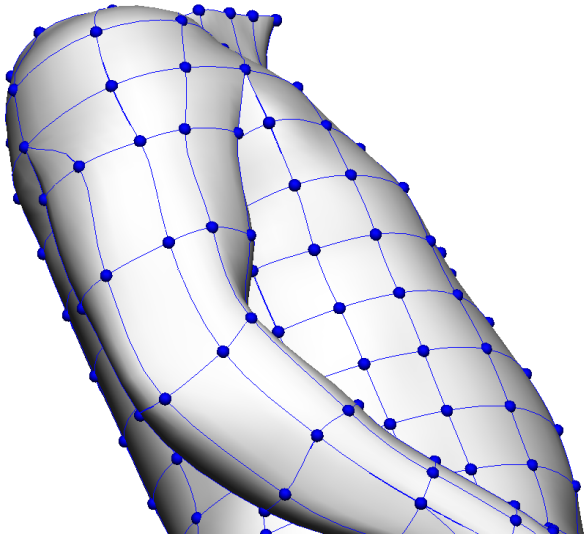
The mesh width of the meshes obtained using the MAP client was chosen such that the surface fitting yielded good results. The meshes are not necessarily ready for use in a simulation, especially if a high mesh resolution is desired. Apart from the mesh width also the type of elements can be different than what is needed for a Finite Element simulation. Our goal is to obtain meshes with linear or quadratic Lagrange elements with configurable mesh widths for the specified upper arm muscles, such as the biceps brachii.

Therefore, the next step of the workflow, as visualized by the right branch of Fig. 1.2, is to transform the volume mesh into a surface mesh which then can be used as basis for further meshing. This is visualized with the example of the biceps muscle in Fig. 1.7. The basis is the Hermite mesh shown in the left-most image.

The Hermite elements can be triangulated and stored as an STL file using the tool `cmgui`. This process triangulates the non-planar faces of all Hermite elements. This leads to a dataset with triangles on the surface and in the inside of the volume, as can be seen in the second image of Fig. 1.7. At this stage, the use of the MAP and OpenCMISS related



(a) Mesh of the trunk and limbs, surface has been triangulated for visualization.



(b) Detail view of part of the right upper arm and the trunk with blue nodes and contours of a mesh with cubic Hermite elements.

Figure 1.6: Mesh of the Visible Human Male from the Visible Human Project.

tools is finished and further processing steps are performed using tools from opendihu that we developed on our own.

A Python script removes the triangles in the inside of the volume. The detection whether a triangle is inside the volume is done by casting four rays from the center of gravity of the triangle and determining if the rays intersect any other triangles. The rays have directions $(x, y, z) = (\pm 1, \pm 1, \frac{1}{3})$, where the z axis is oriented along the muscle and the x and y are oriented in radial direction. The ray-triangle intersection is done using the fast Möller-Trumbore algorithm [Möl97]. For every ray, all triangles are checked. Only if all four rays intersect at least one more triangle, the starting triangle is considered to be inside the volume and subsequently removed from the dataset.

This algorithm has a quadratic time complexity $\mathcal{O}(n^2)$ in the number of triangles n. It could be improved by organizing the triangles in a spatially adaptive data structure, such as an octree. Since this preprocessing step has to be performed only once for a given geometry the runtime is not a concern and there is no need for such optimization.

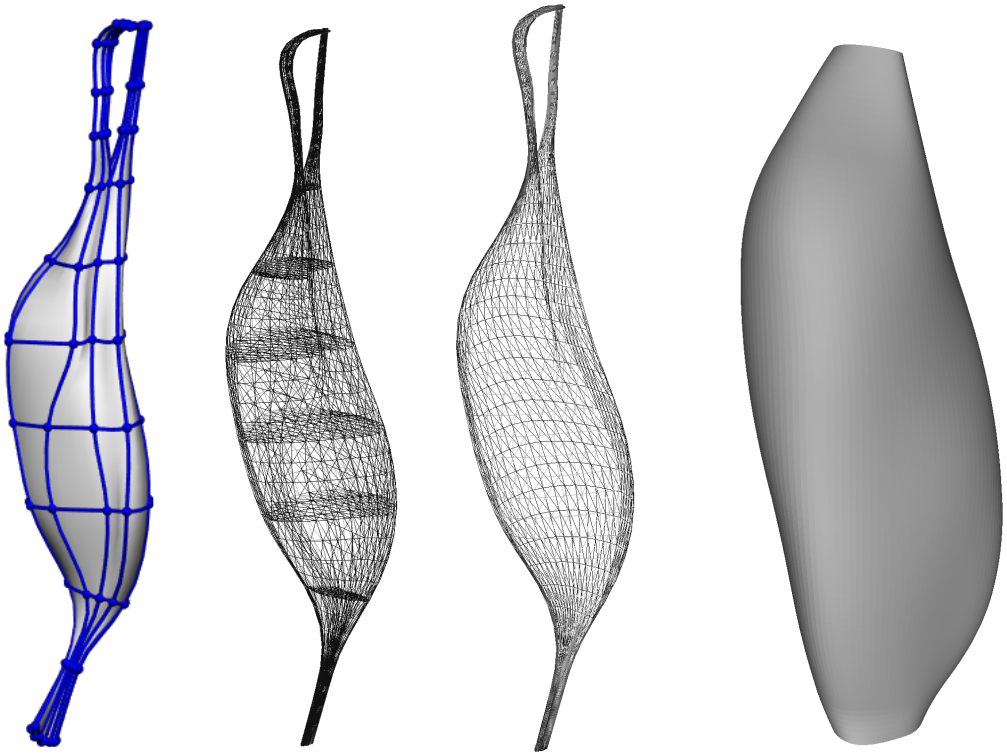


Figure 1.7: Processing the geometry of the biceps brachii muscle. From left to right: mesh with cubic Hermite elements, STL mesh with inside triangles, STL surface mesh where triangles lying inside have been removed, Spline surface of the muscle belly.

The result of this operation is a triangulated surface, shown in the third image of Fig. 1.2. The next step is to create a Spline surface of the muscle belly, as shown in the right-most image of Fig. 1.2. This is described in the next section.

1.2.4 Fitting a Spline surface

The surface representation of the muscle could be obtain from either the left or the right branch of the preprocessing workflow in Fig. 1.2. The surface is given by a point cloud or a number of triangles. To remedy eventual outliers or unphysiological sharp edges from the segmentation, a Spline surface is fitted to the data. This leads to a smooth surface representation and later to a better conditioned Finite Element mesh in the simulation. However, this step is optional. It is also possible to use the surface triangulation from previous section 1.2.3 for the meshing algorithm in the next section 1.3.

Nonuniform rational B-Splines (NURBS) are used for the surface. A NURBS surface is a generalization of a B-Spline surface. From a modeling point of view, B-spline surfaces have

three advantages. First, the B-spline surface can be constructed with given smoothness properties. Second, the definition of a particular B-spline surface builds on geometric information, which simplifies their creation. More specifically a control polygon mesh in the 3D space is defined. Its convex hull is guaranteed to contain the surface. Third, the geometric parameters of a B-spline surface have only local impacts on the shape of the surface. This allows a B-spline surface of a fixed, low polynomical degree to approximate point clouds with any number of points without loosing approximation quality.

A limitation of B-spline surfaces is that circular and spherical shapes cannot be represented. This limitation is overcome by NURBS surfaces. NURBS surfaces are defined as the perspective projection into 3D space of a B-spline surface in 4D space.

The mathematical description is given in this section, following the notation of [Pie12]. The building blocks are the B-spline basis functions of polynomial degree p. Given a knot vector

$$\Xi = (\xi_1, \xi_2, \ldots, \xi_k), \quad \text{with } a = \xi_1 \leq \xi_2 \leq \cdots \leq \xi_k = b,$$

the ith B-spline basis function $N_{i,n}$ of degree n is defined recursively starting with the piecewise constant function $N_{i,0}$ for $n = 0$,

$$N_{i,0}(\xi) = \begin{cases} 1 & \xi_i \leq \xi < \xi_{i+1}, \\ 0 & \text{else}, \end{cases}$$

and using the following relation to define the functions of higher degree, $n > 0$,

$$N_{i,n}(\xi) = \frac{\xi - \xi_i}{\xi_{i+n} - \xi_i} N_{i,n-1}(\xi) + \frac{\xi_{i+n+1} - \xi}{\xi_{i+n+1} - \xi_{i+1}} N_{i+1,n-1}(\xi), \quad i > 0.$$

Because neighbouring entries in the knot vector can be equal, the fraction $0/0$ can occur. In this case, $0/0 := 0$ is defined.

A B-spline curve $\mathbf{C} \in \mathbb{R}^d$ of polynomial degree p is defined as

$$\mathbf{C}(u) = \sum_{i=1}^{\ell} N_{i,p}(u)\, \mathbf{P}_i, \quad u \in [a, b].$$

The coefficients $\mathbf{P}_i \in \mathbb{R}^d, i = 1, \ldots, \ell$ to the basis functions $N_{i,p}$ are called *control points* and define the control polygon. The number ℓ of basis functions and control points is

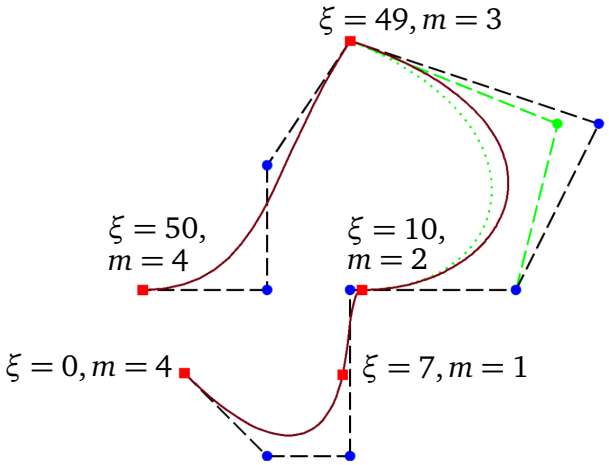


Figure 1.8: Exemplary B-spline curve (red) of degree $p = 3$ for the knot vector $\Xi = (0, 0, 0, 0, 7, 10, 10, 49, 49, 49, 50, 50, 50, 50)$, control points (blue) and control polygon (black). Positions of the curve $\mathbf{C}(\xi_i)$ at the knots ξ_i are indicated by the red squares and the knot value ξ and its multiplicity m is given. The effect of moving one control point is shown in green.

determined from the number of knots k in an open knot vector and the polynomial degree p as $\ell = k - p - 1$.

The number of equal entries in series in the knot vector is the *multiplicity* of the respective knot value. Usually *open* knot vectors Ξ are used where the first and the last knot occur with a multiplicity of $p + 1$. This make the first and last points of the B-spline curve coincide with the control polygon, $\mathbf{C}(a) = \mathbf{P}_1$ and $\mathbf{C}(b) = \mathbf{P}_\ell$.

The multiplicities of the knots in the knot vector encode information about the smoothness of the B-spline curve. If the knot value $\hat{\xi}$ has a multiplicity of m, the B-spline curve will be $p - m$ times continuously differentiable at $\mathbb{C}(\hat{\xi})$.

An exemplary B-spline curve is shown in Fig. 1.8. It uses a *non-uniform* knot vector for polynomial degree $p = 3$, where the differences between neighbouring knot values $\xi_{i+m} - \xi_i$ vary. The effect of different multiplicities can be seen. $m = p = 3$ places the knot on the respective control point, as for $\xi = 49$ in the example. $m = p - 1 = 2$ places the knot on the control polygon, as in the example at $\xi = 10$. A lower multiplicity $m < p - 1$ does not yield to a higher smoothness and in turn does not force the curve on the control polygon. It can also be seen that the B-spline curve stays inside the convex hull of the control polygon which is a property of all B-spline curves [Pie12]. The effect of moving one of the 10 control points is visualized with green color in the Fig. 1.8. The B-spline basis function $N_{i,p}$ has a local support of $S = (\xi_i, \xi_{i+p+1})$. Consequently, only the corresponding part of the curve, $\mathbf{C}(\xi)$ for $\xi \in S$ changes.

A B-spline surface is given as tensor product of two B-spline curves:

$$\mathbf{S}(u,v) = \sum_{i=1}^{\ell^{(1)}} \sum_{j=1}^{\ell^{(2)}} N^{(1)}_{i,p^{(1)}}(u) N^{(2)}_{j,p^{(2)}}(v) \mathbf{P}_{i,j}, \tag{1.1}$$

with two polynomial degrees $p^{(1)}, p^{(2)}$, ansatz functions $N^{(1)}, N^{(2)}$, numbers of ansatz functions $\ell^{(1)}, \ell^{(2)}$ and corresponding knot vectors.

For NURBS, B-spline curves and surfaces are formulated using *homogeneous coordinates*. Every point in Cartesian coordinates $(x,y,z) \in \mathbb{R}^3$ has a set of homogeneous coordinates $(\tilde{x},\tilde{y},\tilde{z},w) = (x\,w, y\,w, z\,w, w)$. Thus, the Cartesian coordinates can be obtain by the *perspective division*, i.e. dividing all but the last coordinate by the weight w.

A NURBS surface is given by the same definition as the B-spline surface in Eq. (1.1) except that the control points $\mathbf{P}_{i,j} \in \mathbb{R}^3$ are enriched with scalar weights $w_{i,j}$ and, thus, replaced by $(\mathbf{P}_{i,j}, w_{i,j}) \in \mathbb{R}^4$. The resulting surface $\mathbf{S}$ is given in homogenous coordinates. Executing the perspective division yields the form:

$$\mathbf{T}(u,v) = \sum_{i=1}^{\ell^{(1)}} \sum_{j=1}^{\ell^{(2)}} R_{i,j}(u,v) \mathbf{P}_{i,j},$$

$$\text{with } R_{i,j}(u,v) = \frac{N_{i,p^{(1)}}(u) N_{j,p^{(2)}}(v) w_{i,j}}{\sum_{r=1}^{\ell^{(1)}} \sum_{s=1}^{\ell^{(2)}} N_{r,p^{(1)}}(u) N_{s,p^{(2)}}(v) w_{r,s}}.$$

The new rational basis functions $R_{i,j}$ and the possibly non-uniform knot vectors give rise to the name non-uniform rational B-spline surface.

In order to find a NURBS surface for the given triangulated surface of a muscle, at first, the part of the geometry corresponding to the tendons is removed) such that the resulting triangles model only the muscle belly. The belly has a length of 12.8 mm.

Then, twelve cross sections are extracted from the surface triangles. The result are twelve horizontal circumference rings. On each ring 9 equidistant points are sampled. The first point is appended after the last point in every ring, such that in total we obtain a grid of 10×12 points. The least squares surface approximation algorithm by [Pie12] is used to fit a NURBS surface to the points. The implementation of the algorithm is given by the NURBS-Python (geomdl) library. Polynomial degrees of $p^{(1)} = 3$ and $p^{(2)} = 2$ are used where the first dimension corresponds to the cross-sectional direction of the muscle. The knot multiplicity is chosen as $m = 1$ for both coordinate directions to obtain

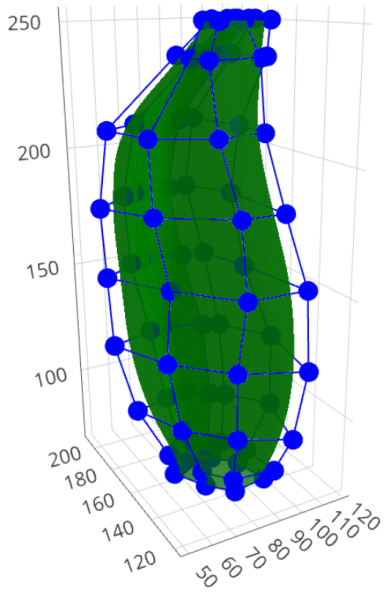
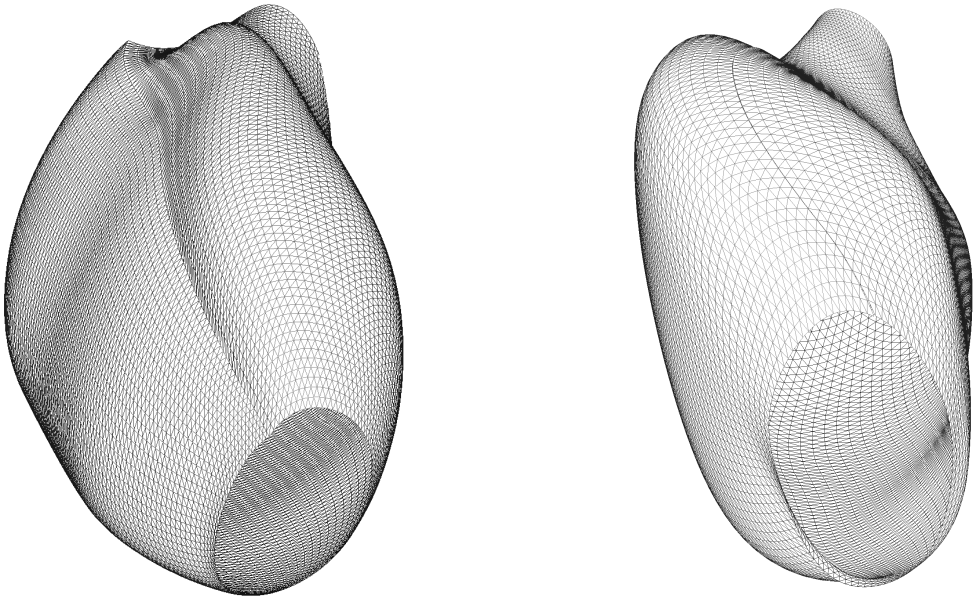


Figure 1.9: Fitted NURBS surface of the biceps muscle (green) and the control polygon (blue).

a two times respective one times continuously differentiable surface in u and v direction. The resulting NURBS surface and the control polygon is visualized in Fig. 1.9. Note that the control polygon is different from the grid of points against which the surface is fitted. Figure 1.10a shows the result of this approach. It can be seen the surface is non-differentiable and has a kink at the seam line where the first and last points of each ring meet. The reason is that the surface fitting algorithm does not pose any conditions on the tangents at the edges of the fitted NURBS surface. Since no implementation of a fitting algorithm specifically for a tubular NURBS surface is available, the point grid is modified. The series of 9 equidistant points on each ring is replicated twice and the first point is again added as the last point. This leads to a grid of $(3 \cdot 9 + 1) = 28 \times 12$ points which wraps around the muscle volume three times. The fitting algorithm is executed on this grid. The resulting NURBS surface also wraps around the muscle three times with the two ends being again not properly fitting to each other. From these three wraps, the middle one is extracted. This corresponds to restricting the NURBS surface $\mathbf{T}(u, v)$ from $(u, v) \in [0, 1]^2$ to $(u, v) \in [0.4, 0.733] \times [0.1]$.

The result is depicted in Fig. 1.10b and it can be seen that the tangents now match very well between the two sides of the NURBS surface. Furthermore, the comparison with the inital approach in Fig. 1.10a shows that an artifical bulge at the top of the muscle in the perspective of the visualization is removed. The overall shape of the muscle now looks smoother and more natural. Also in comparison with the result of the automatic



(a) First approach with 10×12 points. The kink at the seam line along the muscle is clearly visible.

(b) Second, improved approach with 28×12 points. It can be seen that the tangent at the seam line matches very well.

Figure 1.10: Fitted NURBS surface of the biceps muscle, triangulated for visualization purposes.

algorithm in Fig. 1.5a, the results of this approach are smoother.

The generated tubular surface has two holes at the top and bottom which prevent it from being an enclosing surface to the muscle belly volume. The borders of these holes each lie in a plane and, thus, the missing surface is treated as being planar when the 3D meshes are created.

1.3 Serial Algorithm to Create Muscle and Fiber Meshes

Next, a 3D mesh for the muscle volume and 1D meshes for muscle fibers need to be generated from the surface representation described in the last sections. In this section, first an algorithm for the 3D mesh is described. Then, a second algorithm that reuses results from the first algorithm is presented which generated one dimensional meshes for muscle fibers. Both algorithms are executed in serial. A derived algorithm that can run in parallel and, thus, on a distributed memory hardware can handle larger datasets is given in the next section, Sec. 1.4.

The serial algorithm for generation of the 3D mesh is presented in Alg. 1 and the steps visualized in Alg. 1. Input is the set of triangles at the tubular surface of the muscle. The tubular surface is oriented along the z axis. In the following descriptions the muscle in considered to be oriented upright such that z axis points in vertical direction towards the top. The borders at bottom and top have a constant z coordinate.

Algorithm 1 Serial algorithm

```
1 procedure Create_3D_mesh
  Input: triangulated tubular surface
  Output: structured 3D volume mesh

2   Slice geometry
3   Triangulate 2D slices
4   Compute harmonic maps u,v
5   Construct regular grid in parameter space and map to slices
6   Form 3D quadrilateral elements between the 2D slices' meshes
```

1.3.1 Slicing of the geometry

The first step in line 1.2 states to slice the geometry. This means that horizontal "slices" of the cross-sectional area are extracted from the surface mesh. First, the muscle is divided into equidistant positions $z_i, i = 1,\dots,n$ along the z-axis where the slices are to be extracted. As can be seen in Fig. 1.11a, $n = 13$ z coordinates are selected. Next, for every position z_i, all surface triangles T_j that intersect the plane $Z_i = \{\mathbf{p} = (x,y,z) | z = z_i\}$ are considered and the intersection lines $P = T_j \cap Z_i$ are computed. The method of computing plane-triangle intersection is described in the following.

Given is a triangle T with points $\mathbf{p}^1, \mathbf{p}^2, \mathbf{p}^3 \in \mathbb{R}^3$ and a value $\hat{z}$, the wanted result is the set of points $P = T \cap \{\mathbf{p} = (\mathbf{p}_x, \mathbf{p}_y, \mathbf{p}_z) \mid \mathbf{p}_z = \hat{z}\}$ which corresponds to a line segment $\overline{\mathbf{p}^a \mathbf{p}^b}$.

We describe the points in the triangle by two barycentric coordinates ξ_1 and ξ_2 as

$$\begin{aligned} &\mathbf{p}(\xi_1,\xi_2) = (1-\xi_1-\xi_2)\,\mathbf{p}^1 + \xi_1\,\mathbf{p}^2 + \xi_2\,\mathbf{p}^3, \\ &\text{with } \xi_1+\xi_2 \le 1, \quad 0 \le \xi_1, \xi_2 \le 1. \end{aligned} \tag{1.2}$$

By letting $\mathbf{p}_z(\xi_1,\xi_2) = \hat{z}$ we calculate the equation for the line segment $\overline{\mathbf{p}^a \mathbf{p}^b}$ in barycentric

coordinates to be

$$\xi_1 = m \cdot \xi_2 + c, \quad m = -\frac{\mathbf{p}_z^3 - \mathbf{p}_z^1}{\mathbf{p}_z^2 - \mathbf{p}_z^1}, \quad c = \frac{\hat{z} - \mathbf{p}_z^1}{\mathbf{p}_z^2 - \mathbf{p}_z^1}, \quad \mathbf{p}_z^2 \neq \mathbf{p}_z^1.$$

For $\mathbf{p}_z^1 = \mathbf{p}_z^2 \neq \mathbf{p}_z^3$ we swap $\mathbf{p}_z^2$ and $\mathbf{p}_z^3$.

We consider the three sides $\overline{\mathbf{p}^1\mathbf{p}^2}, \overline{\mathbf{p}^2\mathbf{p}^3}$ and $\overline{\mathbf{p}^3\mathbf{p}^1}$ of the triangles and check which of them is intersected by the $z = \hat{z}$ plane.

1. On the triangle side $\overline{\mathbf{p}^1\mathbf{p}^3}$ we have the condition $\xi_2 = 0$ and the side intersects the plane at $\mathbf{p}(0, -c/m)$ iff $0 \leq c \leq 1$.

2. On the triangle side $\overline{\mathbf{p}^1\mathbf{p}^2}$, the condition $\xi_1 = 0$ holds and the side intersects the plane at $\mathbf{p}(c, 0)$ iff $m \neq 0 \wedge 0 \leq -c/m \leq 1$.

3. The third triangle side $\overline{\mathbf{p}^2\mathbf{p}^3}$ is intersected for $\xi_1 = (c+m)/(1+m)$ at $\mathbf{p}(\xi_1, 1-\xi_1)$ iff $m \neq -1 \wedge 0 \leq (c+m)/(1+m) \leq 1$.

Only if exactly two of these three conditions for intersection of the triangle sides are met, there is an intersecting line segment $\overline{\mathbf{p}^a\mathbf{p}^b}$ with $\mathbf{p}^a \neq \mathbf{p}^b$ and the two intersection points $\mathbf{p}^a$ and $\mathbf{p}^b$ on the triangle sides need to be computed. The case $\mathbf{p}^a = \mathbf{p}^b$ and the case where two or more of the triangle points are lying on the $z = \hat{z}$ plane are handled separately.

After the presented computations are performed for all planes Z_i and all triangles T_j, we have a number of line segments that form a geometric "ring" for each z plane. The line segments are ordered according to their adjacency and a counter-clockwise orientation with respect to the z axis is ensured. The length of each ring is computed. A number $m = 16$ of equidistant points is selected on each ring.

Because the selected points on the rings are later used as border points of the volume mesh, their position relative to each other should be in a tidy manner. When viewing the points on the surface in a $x-z$ or $y-z$ projection, they should approximately form a uniform regular grid. With given rings and number m of equidistant points per ring, only the position of one point per ring is not yet fixed. We define a first condition that relates the point positions of two neighbouring rings and a second condition for one point at the bottom-most ring.

The first condition should ensure that the point positions on neighbouring rings are as similar as possible. This is done by minimizing the distance between the one point on every ring that is fixed first. In the algorithm, the z planes are traversed from bottom to

top. The first point $\tilde{p}_{i,0}$ on a ring at $z = z_i$ is determined from the first point $\tilde{p}_{i-1,0}$ of the previous ring at $z = z_{i-1}$ as the one with the minimal distance $|\tilde{p}_{i,0} - \tilde{p}_{i-1,0}|$. Thus, the searched point $\tilde{p}_{i,0}$ has the property that the line between $\tilde{p}_{i,0}$ and $\tilde{p}_{i-1,0}$ and the tangent of the ring are perpendicular.

Given any point $\mathbf{p}$ on the ring at z_i and the tangent vector $\mathbf{u}$ at this point, we can project the connection vector $\mathbf{v}$ from the start point $\tilde{p}_{i-1,0}$ of previous ring to $\mathbf{p}$, $\mathbf{v} = \tilde{p}_{i-1,0} - \mathbf{p}$, onto the tangent $\mathbf{u}$. This leads to the plumb foot point $\mathbf{p}_0$ by the computation

$$\mathbf{p}_0 = \mathbf{p} + t\,\mathbf{u} \quad \text{with } t = \frac{\mathbf{v}\cdot\mathbf{u}}{|\mathbf{u}|^2}.$$

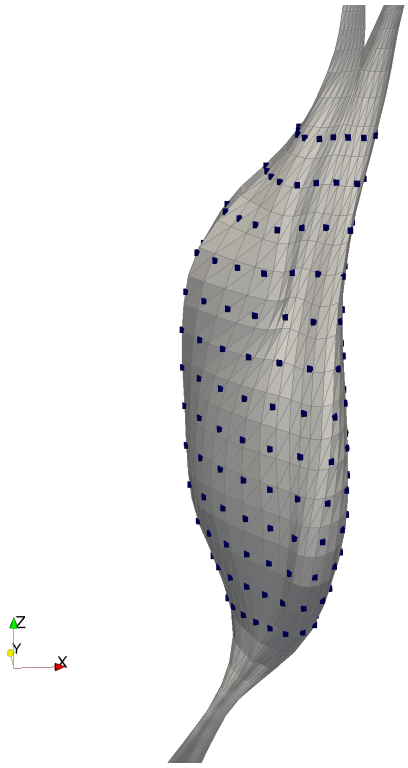
Performing this calculation for every line segment on a ring allows to select the plumb foot $\mathbf{p}_0$ with the smallest distance to the start point $\tilde{p}_{i-1,0}$ of the previous ring to be the start point $\tilde{p}_{i,0}$ of the current ring.

With this first condition, all points are fixed relative to each other. The definition of one point, the start point $\tilde{p}_{0,0}$ of the bottom-most ring, is missing. The second condition therefore fixes this point by a prescribed plane $x = \hat{x}$ and selecting $\tilde{p}_{0,0}$ such that its x coordinate lies in this plane. From the (usually two) points that meet this condition, the one with lower y coordinate is selected. The actual value of $\hat{x}$ is determined experimentally such that the resulting point positions are visually uniform. Because of the shape of the biceps muscle, especially the groove where the humerus bone is located, not every value leads to a good result.

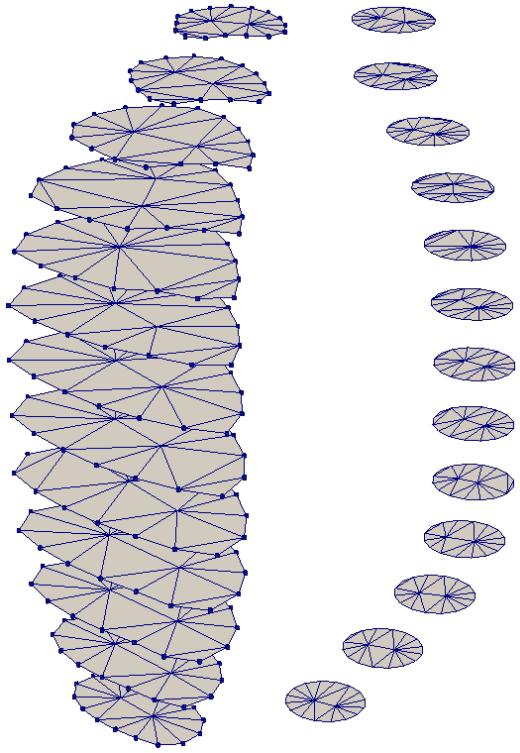
The resulting grid of points on the biceps surface is visualized in Fig. 1.11a. It can be seen that every points of the same ring have the same z coordinate. By connecting neighbouring points, a regular grid can be formed. This overall grid in this $x-z$ perspective view has rather vertical connection lines and looks relatively uniform, compared with the gray surface triangulation mesh of the biceps geometry. The spacing between the points is lower at the top and bottom of the muscle because of the smaller circumference at these locations.

1.3.2 Triangulation of the Slices

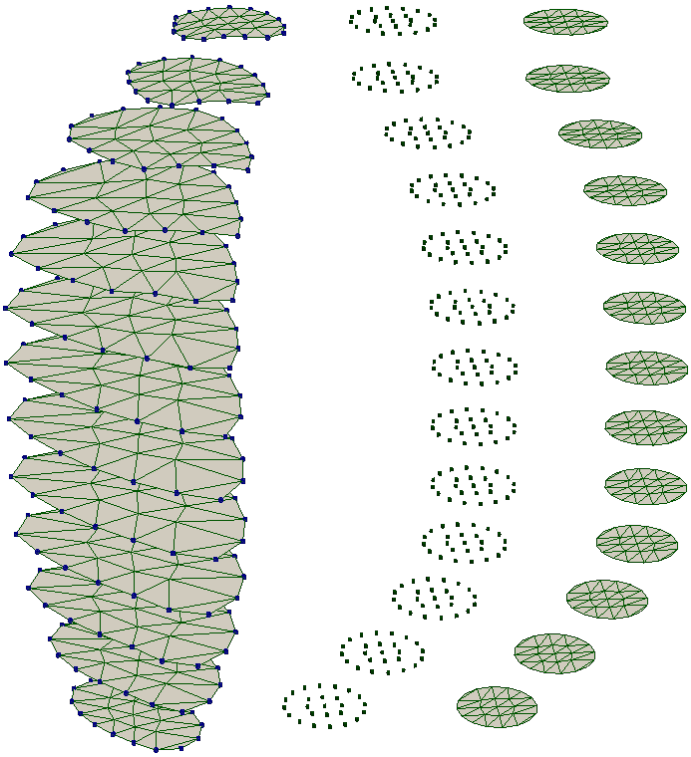
The points of each ring enclose a planar, polygonal surface, a "slice" S_M of the muscle. The next step in the algorithm, line 1.3, is to triangulate the extracted slices, i.e. construct triangles that decompose the polygons. This is visualized at the left side in Fig. 1.11b.



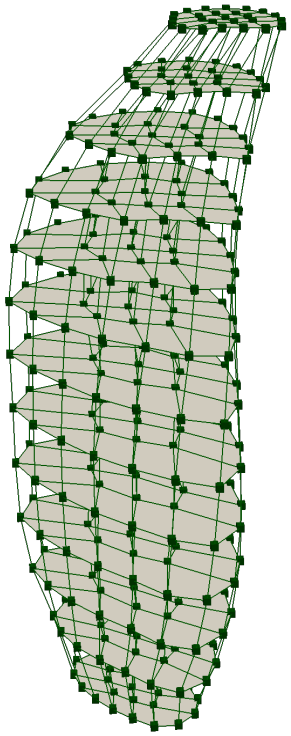
(a) Extracted border points (blue) on the biceps surface mesh (gray). This is the result of line 1.2.



(b) The generated triangulation of the slices (left) and the image $\mathbf{y}(\mathbf{x})$ of the triangulation under the harmonic map (right). This figure shows the result of lines 1.3 and 1.4.



(c) Grid in parameter space (right) and muscle domain (left), result after line 1.5.



(d) 3D elements formed by connecting the slices, result after line 1.6.

Figure 1.11: Steps of the serial algorithm, Alg. 1, executed directly on the surface mesh of the biceps muscle (not the B-spline surface).

Three different methods have been selected to construct the triangulation. The first and second methods are based on Delaunay triangulations, the third method constructs a custom triangulation. Figure 1.12 visualizes results of the three methods.

The first method uses the Delaunay tesselation algorithm from the spatial algorithms and data structures module of the Python package SciPy. The Quickhull algorithm [Bar96] is used which triangulates the convex hull of the points. In consequence, the triangulation of concave slices have triangles that lie outside the interior of the slice. An advantage is that the triangulation uses all given points and no new points are added.

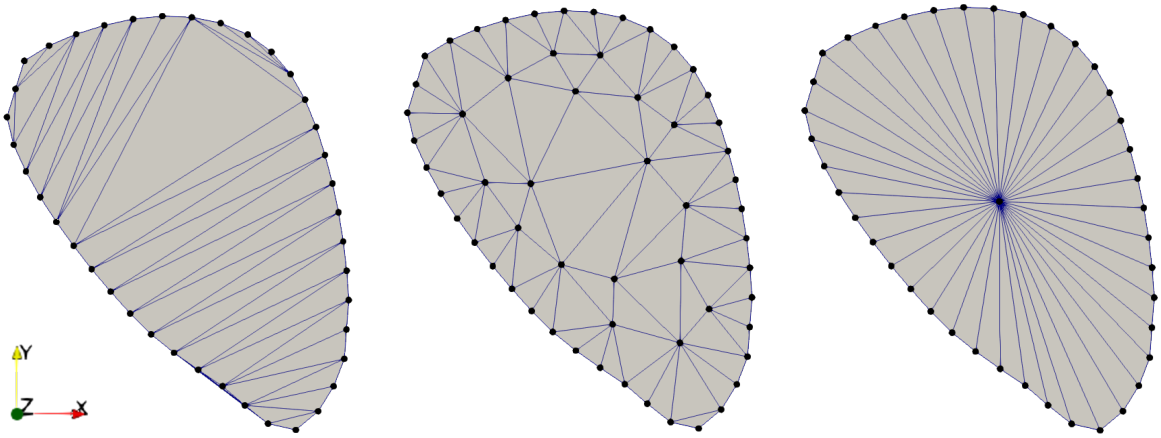
The example in Fig. 1.12a shows a concave slice. At the bottom of the domain, the triangles are outside the slice and almost degenerate.

The second method uses a Delaunay refinement algorithm described in [She02] and implementated in the *Triangle* software [She96]. A conforming, constrained Delaunay triangulation is created. The triangulation correctly handles convex and concave domains. Conforming means that the triangulation uses the given points at the border. Additional points at the border as well as in the interior are added. The triangulation is constraint to generate triangles with minimum angles of 20 degrees and a maximum area A that is set to a value dependend on the area of the bounding box. In consequence, all generated triangulations have a guaranteed mesh quality in terms of angles and about the same size and number of triangles.

Comparing the result of the second method in Fig. 1.12b with the result of the first method in Fig. 1.12a shows the better triangulation quality as the triangles all have higher angles.

The third method places one additional point in the center of gravity of the given points. Triangles are constructed by connecting the center point with two adjacent points on the border, for all given points. The resulting triangulation resembles a pie chart. For some concave slices, this method also creates triangles that partly lie outside the slice. The advantage of this approach is its simplicity.

Figure 1.12c shows the result for an exemplary slice. Here, in contrast to the first method, the third method creates a valid triangulation despite the concave domain.



(a) First triangulation method (b) Second triangulation method (c) Third triangulation method

Figure 1.12: Result of different triangulation methods for a slice in the center of the biceps muscle.

1.3.3 Harmonic Maps

The next step is line 1.4, compute the harmonic maps u and v. For a given slice S_M, these functions map from S_M to coordinates $u, v \in \mathbb{R}$ of a parameter domain $\Omega_P \subset \mathbb{R}^2$. The parameter domain is either a unit circle or a unit square.

The vector $\mathbf{y}(\mathbf{x}) := (u(\mathbf{x}), v(\mathbf{x}))^\top$ for $\mathbf{x} \in S_M$ is interpreted as position in Ω_P. The maps are constructed such that the border ∂S_M of the slice S_M is evenly mapped to the border $\partial \Omega_P$ of the parameter domain Ω_P. The mapping $\mathbf{y} : S_M \to \Omega_P$ is bijective and harmonic, i.e. the Laplace operator of u and v is zero.

More specifically, the definition is

$$\begin{aligned} &u : S_M \to \mathbb{R}, \quad v : S_M \to \mathbb{R}, \\ &\Delta u(\mathbf{y}) = 0, \quad \Delta v(\mathbf{y}) = 0 \quad \forall \mathbf{y} \in S_M. \end{aligned} \tag{1.3}$$

For a uniform parametrization $\mathbf{p} : [0,1] \to \partial S_M$ of the border ∂S_M of the slice, i.e.,

$$\frac{\partial \ell(t)}{\partial t} = c \in \mathbb{R} \quad \forall t \in [0,1], \quad \text{where } \ell(t) := \int_0^t |\mathbf{p}'(\tau)| \, \mathrm{d}\tau,$$

we require the mapping of the parametrization to be also uniform, i.e.,

$$\frac{\partial \ell_P(t)}{\partial t} = c_P \in \mathbb{R} \quad \forall t \in [0,1], \quad \text{where } \ell_P(t) := \int_0^t |\mathbf{y}'(\mathbf{p}(\tau)|\,\mathrm{d}\tau.$$

This leads to the following Dirichlet boundary conditions that close the definition in Eq. (1.3):

$$u(\mathbf{x}_{\mathrm{M,border}}) = u_{\mathrm{P,border}}, \quad v(\mathbf{x}_{\mathrm{M,border}}) = v_{\mathrm{P,border}}. \tag{1.4}$$

Equation (1.3) describes a boundary value problem of ordinary differential equations in u and v. We solve it using the Finite Element Method and the spatial discretization given by the triangulation of the slices.

The first step is to compute the prescribed border points in parameter space, $(u_{\mathrm{P,border}}, v_{\mathrm{P,border}})^\top$. When using the first and third triangulation methods, the border points on the slices are equidistant and therefore the same number of points need to be sampled equidistantly on the border $\partial\Omega_P$ of the parameter space. If the second triangulation method which added additional points is used, also the same number of points are created on the border $\partial\Omega_P$ of the slice as are given on the slice ∂S_M. The border points are created such that the relations of their distances is the same on $\partial\Omega_P$ as for the original points on ∂S_M.

Using the standard procedure of the Finite Element method for $\nabla u(\mathbf{x}) = 0$ on S_M and $u = f(\mathbf{x})$ on ∂S_M, e.g. as outlined in [Rem10], leads to the weak form with ansatz and test functions ϕ,

$$\int_{S_M} (\nabla u^\top \nabla \phi + \nabla f(\mathbf{x})^\top \nabla \phi)\,\mathrm{d}\mathbf{x} = 0 \quad \forall \phi \in \mathcal{H}_0^1. \tag{1.5}$$

Standard linear hat functions are used on the triangles with the interpolation property $\phi_i(\mathbf{x}_j) = \delta_{ij}$. Using the barycentric parametrization of triangles with points $\mathbf{p}^1, \mathbf{p}^2$ and $\mathbf{p}^3$ introduced in Eq. (1.2) we get for the ansatz functions and their derivatives within the elements

$$\phi_1^{(e)} = (1-\xi_1)(1-\xi_2), \qquad \phi_2^{(e)} = \xi_1(1-\xi_2), \qquad \phi_3^{(e)} = (1-\xi_1)\xi_2,$$

$$\nabla\phi_1^{(e)} = (\xi_2 - 1, \xi_1 - 1)^\top, \qquad \nabla\phi_2^{(e)} = (1-\xi_2, -\xi_1)^\top, \qquad \nabla\phi_3^{(e)} = (-\xi_2, 1-\xi_1)^\top.$$

The superscript $\square^{(e)}$ refers to the definition of the functions within elements. The usual global assembly involves composing the global, nodal functions $\phi_i(\mathbf{x})$ for nodes indexed by $i = 1, \ldots, n_{\text{nodes}}$ and using a mapping between the barycentric coordinates $\xi_1, \xi_2 \in [0,1]^2$ inside the elements to the global coordinates $\mathbf{x} \in S_M$.

Inserting the discretization

$$u_h(\mathbf{x}) = \sum_{i=1}^{n_{\text{nodes}}} u_i \, \phi_i(\xi_1(\mathbf{x}), \xi_2(\mathbf{x}))$$

into Eq. (1.5) leads to the form

$$\sum_{i=1}^{n_{\text{nodes}}} u_i \int_{S_M} \nabla_{\mathbf{x}} \phi_i^\top \nabla_{\mathbf{x}} \phi_j \, \mathrm{d}\mathbf{x} + \sum_{i=1}^{n_{\text{nodes}}} f_i \int_{S_M} \nabla_{\mathbf{x}} \phi_i^\top \nabla_{\mathbf{x}} \phi_j \, \mathrm{d}\mathbf{x} = 0 \quad \forall j = 1, \ldots, n_{\text{nodes}}. \tag{1.6}$$

The integrations are executed element-wise and over the elemental coordinates ξ_1, ξ_2.

The transformation to elemental coordinates involves the computation of the Jacobian $J = \mathrm{d}\mathbf{x}/\mathrm{d}\boldsymbol{\xi}$ of the mapping between element coordinates $\boldsymbol{\xi} = (\xi_1, \xi_2)$ and global coordinates $\mathbf{x}$. From the definition Eq. (1.2) it follows that

$$J = \frac{\mathrm{d}\mathbf{x}}{\mathrm{d}\boldsymbol{\xi}} = [\mathbf{p}^2 - \mathbf{p}^1, \mathbf{p}^3 - \mathbf{p}^1].$$

The metric tensor for this mapping is given by

$$\mathcal{M} := \left(\frac{\mathrm{d}x}{\mathrm{d}\xi}\right)^T \left(\frac{\mathrm{d}x}{\mathrm{d}\xi}\right).$$

The transformation of the integrals in Eq. (1.6) introduces an additional integration factor of $\sqrt{\det \mathcal{M}}$.

We get the following matrix equation:

$$M_u \mathbf{u} = -M_f \, \mathbf{f}, \tag{1.7}$$

with the vector $\mathbf{u}$ of nodal solution values, the vector $\mathbf{f}$ of nodal Dirichlet boundary condition values at the border and the global stiffness matrices M_u and M_f. The two global stiffness matrices are assembled from the element stiffness matrices M^{el} for the

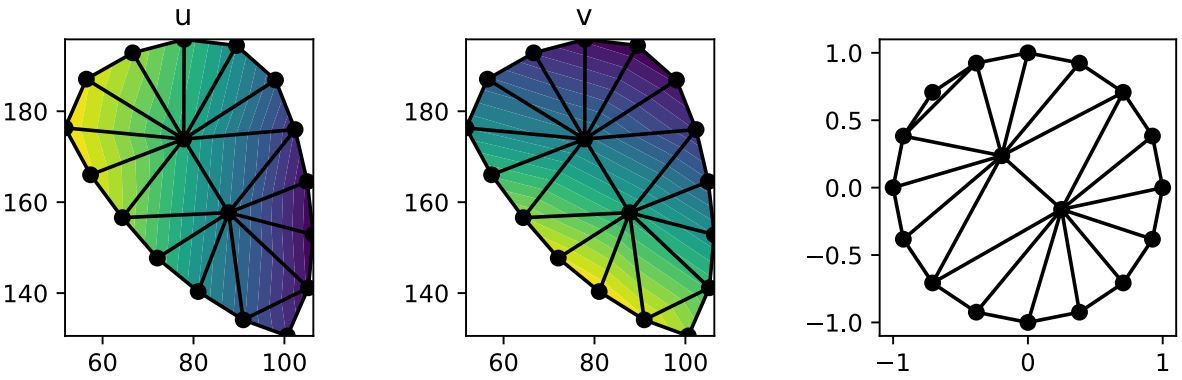


Figure 1.13: Triangulations and harmonic map for a slice S_M of the biceps muscle. The first two plots show the solutions of u and v on the slice S_M. The third plot shows the image in Ω_P of the triangulation in S_M under the harmonic map.

degrees of freedom at all nodes respectively at the border nodes. The entries of the element stiffness matrices are given by

$$M_{i,j}^{\text{el}} = \int_0^1 \int_0^{1-\xi_1} \nabla \phi_i^{(e)}(\xi_1, \xi_2)^\top \mathcal{M}^{-1} \nabla \phi_j^{(e)}(\xi_1, \xi_2) \sqrt{\det \mathcal{M}}\, \mathrm{d}\xi_2 \mathrm{d}\xi_1.$$

By solving Eq. (1.7) for $\mathbf{u}$ we get the discretized harmonic map u. The Finite Element formulation and computation for v is analogous and uses the same global stiffness matrices.

Figure 1.13 visualizes the triangulation of S_M and the solutions $u(\mathbf{x})$ and $v(\mathbf{x})$ in the first two plots. The color range from bright yellow to dark violet corresponds increasing values of u and v. It can be seen that the u values increase from left to right whereas the v values increase from bottom to top, corresponding to the horizontal and vertical coordinate axis y_1 and y_2 of Ω_P.

Applying the computed harmonic map $\mathbf{y}(\mathbf{x})$ to the triangulation of the slices results in a triangulation of the parameter domain Ω_P. This is shown in the third plot of Fig. 1.13 and in Fig. 1.11b. The generated triangulation of the slices was generated using the second triangulation method with the constrained Delaunay triangulation. On the right, the image $\mathbf{y}(\mathbf{x})$ under the harmonic map of the triangulation in the slices is shown on the unit circle parameter domains.

1.3.4 Construction of a regular grid in the parameter domain

The next step in the Alg. 1 is the construction of a structured, regular grid in the parameter domain Ω_P, as stated in line 1.5. This grid will then be mapped to the slices S_M. Creating a structured grid in a given domain is also called *quadrangulation*.

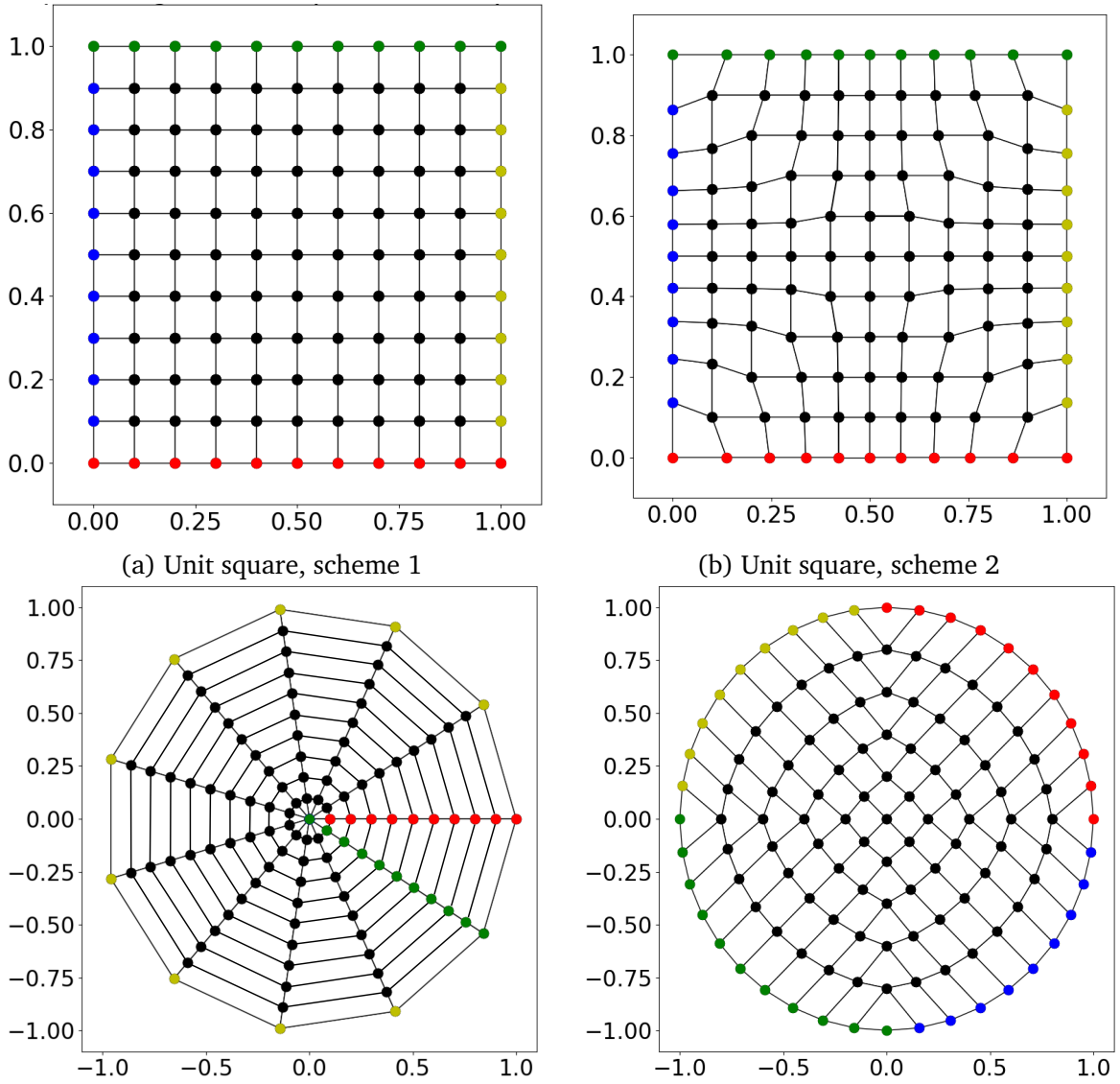
The parameter domain Ω_P can be selected to be either a unit square or a unit circle. For both choices, two different schemes how to generate a grid with given number of cells can be selected. Figure 1.14 shows all four possiblities.

The first scheme, Fig. 1.14a, uses an equidistant regular grid. The grid will be mapped to a cross section of the muscle which has no sharp corners like the unit square. Therefore, the cells of the grid will be distorted at the corners, usually shortening diagonals that point towards the corners and stretching the other diagonals. This assumption motivates the second scheme in Fig. 1.14b. Here, the elements are already distorted in the described manner, with increasing distortion closer to the corners.

A different approach is to use a unit circle, which has no corners and therefore might more resemble a muscle cross section. The first scheme of the unit circle is given in Fig. 1.14c. It uses the radial and circumferential directions for the two dimensions of the grid. A disadvantage of this scheme is that the quadrilaterals at the center are degenerated to triangles. Additionally, the area of the cells varies significantly and the outer cells have unequal side lengths. To remedy this problem, the second scheme given in Fig. 1.14d is constructed. When traversing from the outer border towards the center point and considering the circumferential lines of grid points, the circle morphs into a square as the number of grid points decreases. This approach has the disadvantage that some cells have an inner angle of nearly 180°, especially four elements at the border.

Each construction schemes allows to specify the (squared) number of nodes and in consequence the number of cells. The examples in Figures 1.14a, 1.14b, and 1.14d have 11×11 nodes and 10×10 cells. For Fig. 1.14c, the numbers are slightly different. There are $10 \times 11 + 1$ nodes resulting in 10×11 cells.

Next, the grid in the parameter domain is transferred to the muscle domain by applying the harmonic map $\mathbf{y}(\mathbf{x}) \in S_M$ on every point of the quadrangulation $\mathbf{x} \in \Omega_P$. This is illustrated in Fig. 1.11c for a parameter domain consisting of the unit circle, with quadrangulation scheme 2 and 5×5 nodes. The cells of the grid in Ω_P are shown in the right-most stack of domains. The grid points are visualized left of the grids. The resulting image of the mesh in the slices S_M is shown on the left. For visualization reasons, each quadrilateral has been split into two triangles.



(a) Unit square, scheme 1

(b) Unit square, scheme 2

(c) Unit circle, scheme 1

(d) Unit circle, scheme 2

Figure 1.14: Different quadrangulation schemes of the parameter domain with 11 × 11 nodes. The borders of the grid are colored for better perceptiblity.

1.3.5 Formation of Three-Dimensional Elements

The result of the previous steps is a number of quadrangulated muscle slices. The grid on every slice has the same number of nodes and elements. The nodes on the border of neighbouring slices are positioned similarly.

The final step of Alg. 1 is line 1.6, the formation of 3D elements. Inserting edges between all corresponding nodes on two neighbouring slices creates a set of 3D elements and, thus, an overall 3D quadrilateral mesh of the muscle volume. This step is visualized in Fig. 1.11d.

1.3.6 Generation of Fiber Meshes

Algorithm 2 Serial algorithm

```
1 procedure Create_1D_meshes
  Input: structured 3D volume mesh
  Output: 1D fiber meshes

2   Solve Laplacian flow problem
3   Trace streamlines in the gradient field
4   Sample 1D fiber meshes
```

1.3.7 Results and Discussion

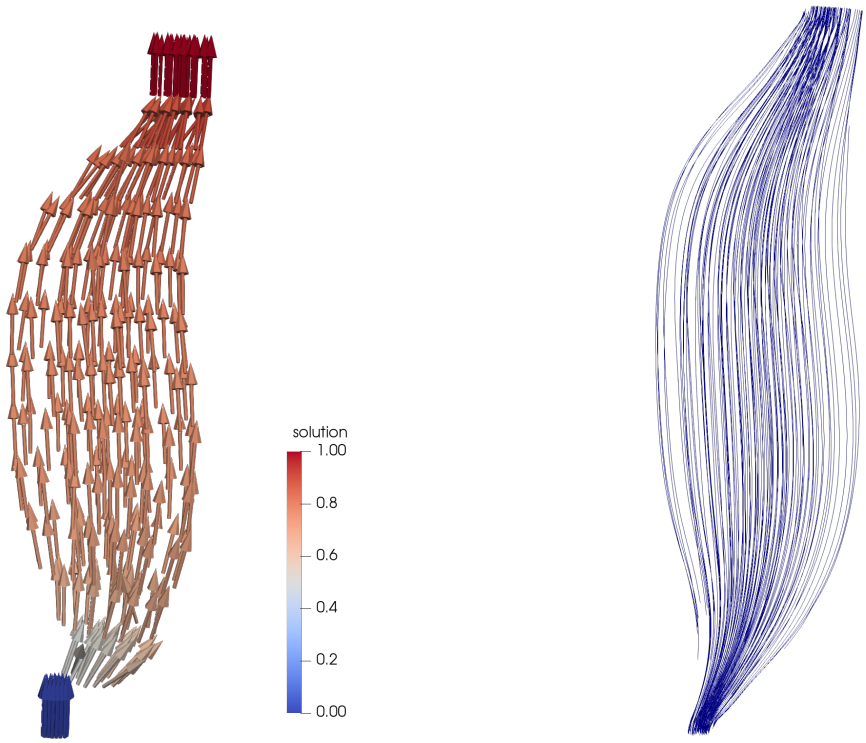
1.4 Parallel Algorithm to Create Muscle and Fiber Meshes

parallel algorihm

1.5 Results and Discussion

conclusion for automatic segmentation: doesn't work for muscles, only for bones

If the quality of the images should be improved, it is also possible to manually edit the intermediate results of segmentations.



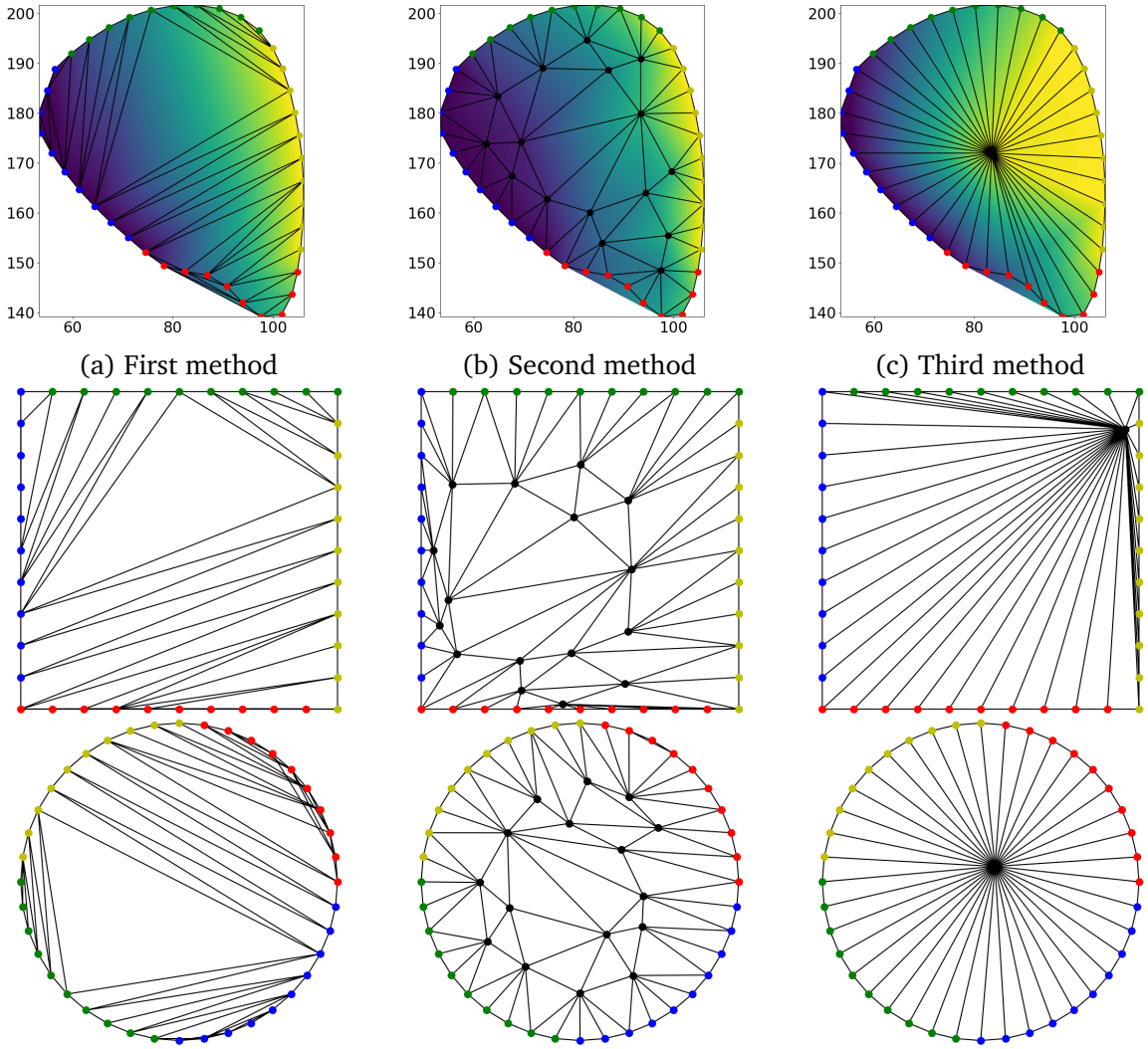
(a) (b)

Figure 1.15: .

Algorithm 3 Parallel algorithm

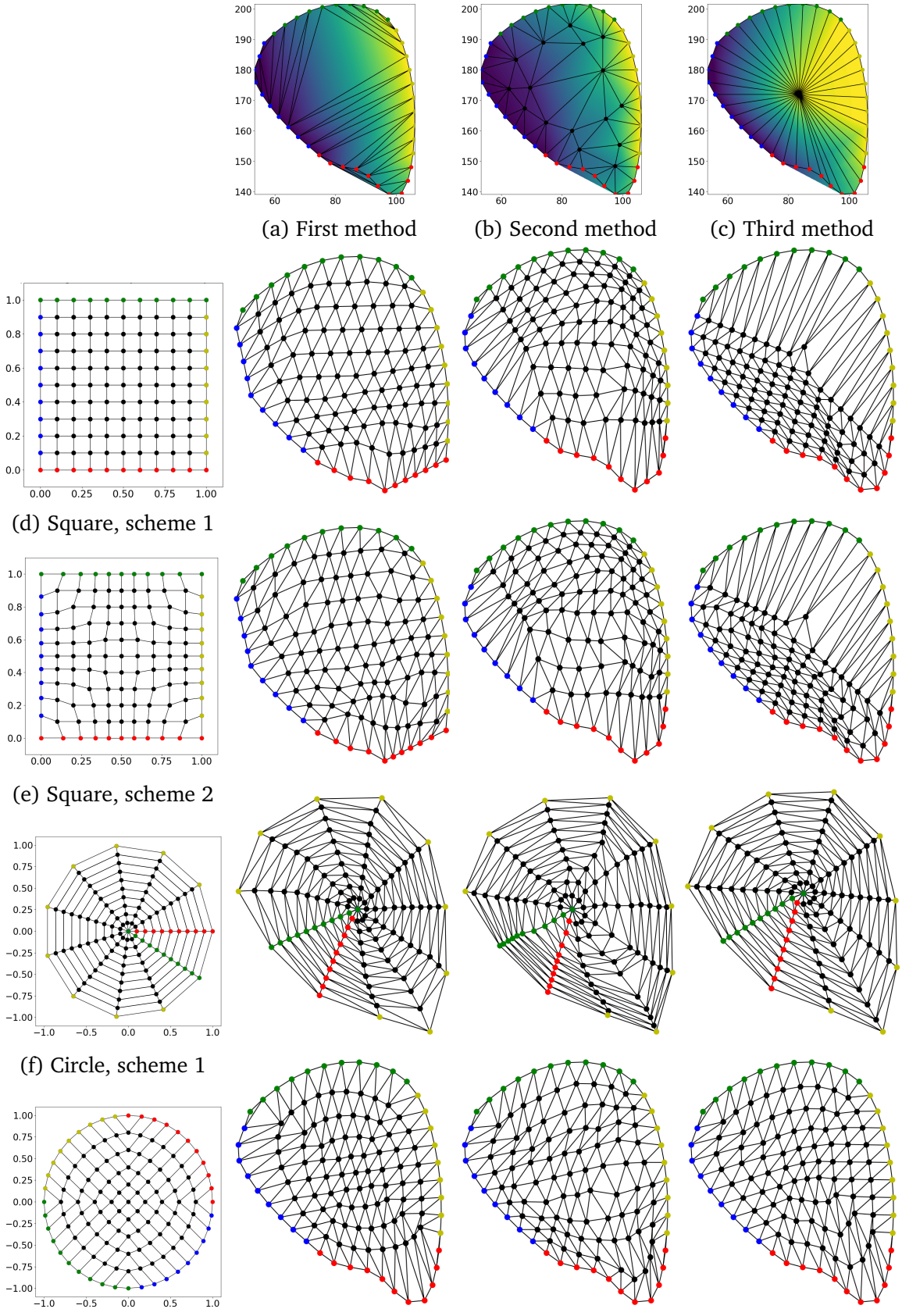
```
1 procedure Create_3D_meshes_parallel
  Input: triangulated tubular surface
  Output: structured 3D volume mesh
  Output: 1D fiber meshes

2   Solve Laplacian flow problem
3   Trace streamlines in the gradient field
4   Sample 1D fiber meshes
```



(a) First method (b) Second method (c) Third method

Figure 1.16: Top row: Different triangulation methods for S_M, the color represents the solution u of the harmonic map. Middle and bottom row: triangulation mapped to the parameter domain Ω_P, for the unit square (middle) and the unit circle (bottom). Each column corresponds to one triangulation method.



(a) First method (b) Second method (c) Third method

(d) Square, scheme 1

(e) Square, scheme 2

(f) Circle, scheme 1

(g) Circle, scheme 2

Figure 1.17: Meshes in S_M for quadrangulations (rows) and triangulations (columns).

1.6 Motor unit distribution

$$p_{\text{unscaled}}(x) = \alpha^x, \quad \text{with } \alpha = 1.2$$

$$p(x) = \frac{p_{\text{unscaled}}(x)}{\int_1^{n_{\text{MU}}} p_{\text{unscaled}}(x)\,\mathrm{d}x}$$

$$c(x) = \int_1^x p(x)\,\mathrm{d}x$$

$c^{-1}(x) draw\_sampℓe() : returnround(c^{-1}(Y)), Y \sim N\big(c(0.5, n\text{MU} + 0.5)\big)$ Zufallsvariable$p_{\text{distance,unscale}}$

Abbreviations

Nomenclature

Bibliography

[Bar96] **Barber**, C. B.; **Dobkin**, D. P.; **Huhdanpaa**, H.: *The quickhull algorithm for convex hulls*, ACM Trans. Math. Softw. 22.4, 1996, pp. 469–483, ISSN: 0098-3500, doi:10.1145/235815.235821, https://doi.org/10.1145/235815.235821

[Kus19] **Kusterer**, J.: *Extraktion anatomischer Strukturen und Darstellung durch NURBS*, Deutsch, Bachelorarbeit: Universität Stuttgart, Institut für Parallele und Verteilte Systeme, Simulation großer Systeme, Bachelorarbeit, 2019, http://www2.informatik.uni-stuttgart.de/cgi-bin/NCSTRL/NCSTRL_view.pl?id=BCLR-2019-19&engl=0

[Möl97] **Möller**, T.; **Trumbore**, B.: *Fast, minimum storage ray-triangle intersection*, Journal of Graphics Tools 2.1, 1997, pp. 21–28, doi:10.1080/10867651.1997.10487468, eprint: https://doi.org/10.1080/10867651.1997.10487468, https://doi.org/10.1080/10867651.1997.10487468

[Pie12] **Piegl**, L.; **Tiller**, W.: *The NURBS book*, Springer Science & Business Media, 2012

[Rem10] **Remacle**, J.-F. et al.: *High-quality surface remeshing using harmonic maps*, International Journal for Numerical Methods in Engineering 83.4, 2010, pp. 403–425, doi:10.1002/nme.2824, eprint: https://onlinelibrary.wiley.com/doi/pdf/10.1002/nme.2824, https://onlinelibrary.wiley.com/doi/abs/10.1002/nme.2824

[She02] **Shewchuk**, J. R.: *Delaunay refinement algorithms for triangular mesh generation*, Computational Geometry 22.1, 2002, 16th ACM Symposium on Computational Geometry, pp. 21–74, ISSN: 0925-7721, doi:https://doi.org/10.1016/S0925-7721(01)00047-5, http://www.sciencedirect.com/science/article/pii/S0925772101000475

[She96] **Shewchuk**, J. R.: *Triangle: engineering a 2D quality mesh generator and delaunay triangulator*, Applied Computational Geometry: Towards Geometric Engineering, ed. by **Lin**, M. C.; **Manocha**, D., vol. 1148, Lecture Notes in Computer Science, From the First ACM Workshop on Applied Computational Geometry, Springer-Verlag, 1996, pp. 203–222

[Spi96] **Spitzer**, V. et al.: *The Visible Human Male: A Technical Report*, Journal of the American Medical Informatics Association 3.2, 1996, pp. 118–130, ISSN: 1067-5027, doi:10.1136/jamia.1996.96236280, eprint: https://academic.oup.com/jamia/article-pdf/3/2/118/2089636/3-2-118.pdf, https://doi.org/10.1136/jamia.1996.96236280

[Zha14] **Zhang**, J. et al.: *The map client: user-friendly musculoskeletal modelling workflows*, Biomedical Simulation, ed. by **Bello**, F.; **Cotin**, S., Cham: Springer International Publishing, 2014, pp. 182–192, isbn:978-3-319-12057-7