A Framework for Modelling Thin Objects: from Meshing to Shell Finite Element Method

No Author Given

No Institute Given

Abstract.

1 Introduction

Human body is composed of various deformable anatomical structures. Realistic modeling of organ deformations is a challenging research field that leads the way to numerous clinical applications including: medical training and rehearsal systems, patient-specific planning of surgical procedure and per-operative guidance based on simulation. In all these cases, the clinician needs fast update of the deformation model to obtain a real-time display of the computed deformations. To realize this kind of interactive systems, one face the trade-off problem between accuracy of the model and the need of fast computations.

An other key challenge of real-time human soft-tissue modeling is the variousness of the mechanical behaviors. It seems unrealistic to target a unique model for all tissues even if there are obvious advantages to make maximum use of optimized generic models. Yet most of previous work focus on volume models that are able to capture the behavior of volume organs like liver [] or brain tissues []. In contrast, this paper seeks to propose a solution for simulating, in real-time, deformation of thin anatomical structures, whose volume is negligible compared to their surface area. Examples of such anatomical structures include hollow structures, like the wall of blood vessels or the colon but also the envelope of volume organs, like the capsule of Glisson.

In the following of this paper, we present a framework for realistic and real-time simulation of thin organs. Our method, presented in section 2 is based on shell theory and computes the internal forces using Finite Element Modeling (FEM) on curved triangles. The design of the elements relies on a polynomial interpolation that is also use to capture the interactions with other models, including volume models. We also present an automatic process to obtain meshes from image based reconstruction in an attempt to optimize the computation using fewer elements (see section 3). The benefits of our approach (meshing of a curved surface, fast computation and possible interactions with volume models) are illustrated using various examples presented in section 4.

2 A physical model for thin structures

2.1 Related work

Development of a satisfactory physical model that runs in real-time but produces visually convincing animation of thin objects is challenging. Rather than resorting to shell theory which involves the most complex formulations in continuum mechanics, previous works have often relied on discrete formulations. Early approaches only considered in-plane deformation, and often relied on mass-spring models. More recent works have considered adding bending through angular springs. For instance Wang et al. [**?**] successfully used a network of linear and angular springs to describe bending and twisting of catheters and guidewires in an interventional radiology simulator. Yet, such models are limited in their ability to describe certain behaviour, as they do not rely on continuum mechanics. Another limitation of such models is the difficulty to derive spring stiffness (in particular for angular springs) from elastic properties (Young's modulus and Poisson's ratio). For these reasons, other approaches have been proposed.

Continuum mechanics provides many formulations able to accurately describe stresses occurring within thin objects. Most of them fall into one of the following two categories: plate theory or shell theory. Those theories have been a subject of interest in the mechanical community for decades. The difference between these two kinds of structures is very well explained by Liu et al. [**?**] and can be summarized by the fact that plate bending elements can only carry transversal loads while shells can undergo more complex deformations. Considering the wide range of shapes and solicitations that can be applied to hollow organs, a thin plate model would not correctly capture their deformation and therefore a shell formulation was retained.

2.2 Co-rotational triangular shell model

A complete description of our co-rotational triangular shell finite element model is available in one of our previous publication [**?**]. Therefore we will only remind briefly the key points. We improved and extended a plate model first introduced by Przemieniecki [**?**] to a co-rotational formulation. Co-rotational approaches offer a good trade-off between computational efficiency and accuracy by allowing small deformations but large displacements. Once combined with an in-plane membrane formulation we obtain an accurate, yet computationally efficient, shell finite element method featuring both membrane and bending energies.

Triangular elastic membrane The element stiffness matrix $\mathbf{K}_e$ can be computed as follows:

$$\mathbf{K}_e = \int_v \mathbf{J}\boldsymbol{\chi}\mathbf{J}^T dV \tag{1}$$

where $\mathbf{J}$ is the strain-displacement matrix and $\boldsymbol{\chi}$ embodies the material's behaviour. The stiffness matrix in the global frame is eventually obtained using

the rotation matrix of the element: $\mathbf{K} = \mathbf{R}^T \mathbf{K}_e \mathbf{R}$ where $\mathbf{R}$ describes the rotation of the (triangular) element with respect to its initial configuration.

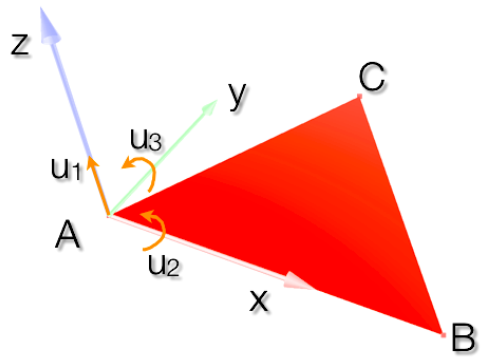

Fig. 1. The different degrees of freedom u of a triangular thin plate in bending are illustrated above.

Triangular plate bending To calculate the stiffness matrix for the transverse deflections and rotations shown on figure 1, we start with the assumed deflection u_z of the form

$$u_z = c_1 + c_2 x + c_3 y + c_4 x^2 + c_5 xy + c_6 y^2 + c_7 x^3 + c_8 xy^2 + c_9 y^3 \tag{2}$$

where $c_1, \ldots, c_9$ are constants. Let us define the vector $\mathbf{u} = \{u_1 u_2 \ldots u_9\}$ of the displacements and slopes at the three corners of the triangular plate using the following notations:

$$u_1 = (u_z)_{x_1,y_1} \qquad u_2 = \left(\frac{\partial u_z}{\partial y}\right)_{x_1,y_1} \qquad u_3 = -\left(\frac{\partial u_z}{\partial x}\right)_{x_1,y_1} \tag{3}$$

and so on for the two other vertices. We then can derive a matrix C such as $\mathbf{u} = \mathbf{Cc}$ where $\mathbf{c} = \{c_1 c_2 \ldots c_9\}$.

We can calculate the strains from the flat-plate theory using:

$$e_{xx} = -z\frac{\partial^2 u_z}{\partial x^2} \tag{4}$$

$$e_{yy} = -z\frac{\partial^2 u_z}{\partial y^2} \tag{5}$$

$$e_{xy} = -2z\frac{\partial^2 u_z}{\partial x \partial y} \tag{6}$$

Symbolically this may be expressed as $\mathbf{e} = \mathbf{Dc}$ where $\mathbf{D}$ derives from equation 2 and the above relations. Noting that $\mathbf{c} = \mathbf{C}^{-1}\mathbf{u}$, we have:

$$\mathbf{e} = \mathbf{DC}^{-1}\mathbf{u} = \mathbf{bu} \tag{7}$$

where the strain-displacement matrix $\mathbf{b} = \mathbf{DC}^{-1}$. The stiffness matrix $\mathbf{K}_e$ for an element is then classically obtained from

$$\mathbf{K}_e = \int_v \mathbf{b}^T \boldsymbol{\chi} \mathbf{b} dV \tag{8}$$

where $\boldsymbol{\chi}$ is the material matrix. The stiffness matrix in the global frame is eventually obtained using the rotation matrix of the element as in the membrane case.

3 Meshing a curved surface

3.1 Related work

A key component of finite element methods is the mesh of the simulated object being used. While a sufficient number of elements is needed to accurately describe the geometry, too many of them will yield a computationally expensive simulation and solving the system in real-time may no longer be possible. Literature about volumic mesh generation algorithms is fairly dense. However, there are only a few that are concerned about the generation of meshes over curved surfaces. One of the most widely used techniques for the creation of surface meshes is the plane to surface transformation method [**?**], mesh is first generated on a two-dimensional domain and then mapped onto the surface. If this method gives reasonably good meshes on smooth surfaces, the results are usually quite poor with more complex curved surfaces. The finite elements may be generated directly on the curved surfaces based on the advancing front technique [**?**,**?**]. The main issue with this approach is that an analytical description of the geometry is needed, which is not the case in medical simulation. Another method consists of using an a priori error estimator to build an adaptive mesh generation [**?**]. However, the tolerance of this indicator should be chosen depending on the desired accuracy of the finite element solution, and therefore requires some knowledge about the problem in order to choose an effective tolerance. In [**?**] the authors start from an existing triangular mesh created with a CAD software and refine and smooth the mesh based on element quality and surface curvature. While all those methods allow their authors to get satisfactory meshes over curved surfaces according to their needs, they all make use of flat elements to mesh geometries and do not generate actual shells.

3.2 A mechanically-driven method to mesh curved surfaces with shells

Indeed, one main difference between shell and plate elements is that the rest shape of the former can describe a curved surface. We propose to take this

crucial feature into account to optimise the meshing process. While many flat triangles are often required to describe highly curved surfaces, fewer triangular shell elements are needed to describe the given geometry with the same precision since they can be naturally curved. In the following we assume that we have a high resolution triangular mesh obtained from a binary segmented image of the organ we want to simulate (via a Marching Cube algorithm for instance). Our goal is to create a mesh featuring the optimal number of shell elements while staying as close as possible to our targeted geometry.

Therefore we need to insure that the distance between the surface of our shell-based mesh and the targeted high resolution mesh will be minimal. To our knowledge the only tool available to measure error between two surfaces is the Hausdorff distance [**?**,**?**]. As a reminder the Hausdorff distance between two meshes is the maximum between the two so-called one-sided Hausdorff distances:

$$d_{\mathrm{H}}(X,Y) = \max\left\{ \sup_{x\in X} \inf_{y\in Y} d(x,y), \sup_{y\in Y} \inf_{x\in X} d(x,y) \right\} \quad . \tag{9}$$

where $d()$ is the Euclidian distance between two points. Since the curved surface of each shell must be taken into account in the computation of the Hausdorff distance, a finer mesh is created by adding vertices on the surface of each element. We first sample the flat surface of each element by recursively dividing each triangle into four smaller ones and the deflection of each new vertex is computed using (2). We then use this new mesh that samples the actual surface described by the shells to compute the Hausdorff distance with the targeted high resolution mesh. The same process of subdivision allows us to render each shell as a curved triangle (Fig. 2).

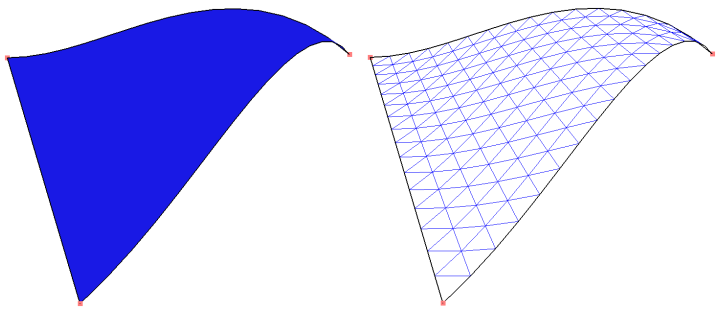

Fig. 2. The triangle formed by the three vertices of the shell has been recursively subdivided 3 times and the deflection of each new vertex was computed according to the same deflection function used in our shell finite element formulation. Sampling the actual surface of the shell allows rendering and more accurate Hausdorff distance computation.

The first step in the process of generating a shell-based mesh is an aggressive decimation of the targeted high resolution mesh. We then apply a heuristic

method derived from the work of Saupin et. al [**?**] with tetrahedral meshes based on very simple geometrical rules. For each node of the coarse mesh, we find the three closest triangles on the high resolution mesh and we move the node to the barycenter of the three centres of mass of those triangles. This technique locally smooths the surface of the mesh while converging towards the desired high resolution mesh. At each iteration of this algorithm we measure the error with the target using the Hausdorff distance and the process is stopped when the required precision has been reached. A simple example is shown Fig. 3 and 4 to illustrate the method.

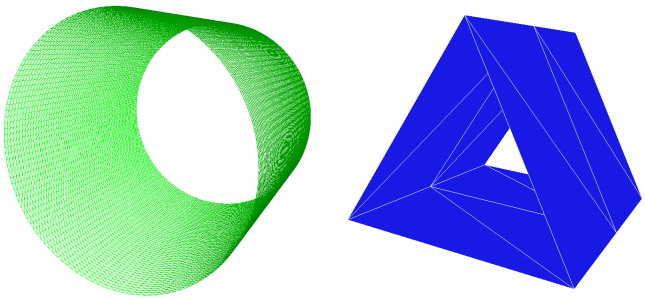

Fig. 3. Illustration of our method on a simple example. The target is the high resolution cylinder mesh (left) and the start point is a very coarse mesh approximating the shape of a cylinder (right).

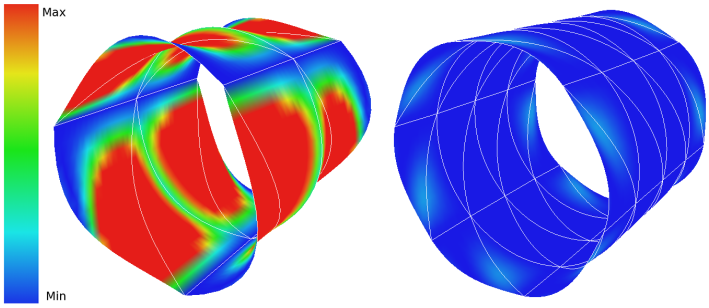

Fig. 4. (Left) Hausdorff distance colour map of the coarse mesh rendered with shells. (Right) Hausdorff distance colour map after one iteration of our algorithm. We can notice that differences with the target mesh have been subtantially reduced.

4 Results

4.1 Meshing of anatomical structures (error maps)

4.2 Deformations

Computation times on 3 examples

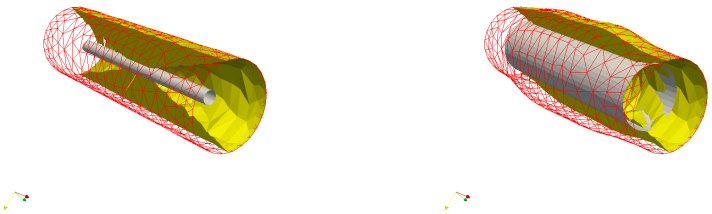

Fig. 5. Avant... Apres...

Coupling between tetrahedra and shells for more advanced modelling

5 Conclusion

Acknowledgments