Workshop on Virtual Reality Interaction and Physical Simulation VRIPHYS (2012)
J. Bender, A. Kuijper, D. W. Fellner, and É. Guérin (Editors)

Bézier Shell Finite Element for Interactive Surgical Simulation

Tomáš Golembiovský[1,2] and Christian Duriez[2]

[1]Faculty of Informatics, Masaryk University, Czech Republic
[2]INRIA Lille – Nord Europe and University of Lille, France

Abstract

There is a strong need, in surgical simulations, for physically based deformable model of thin or hollow structures. The use of shell theory allows to have a well-founded formulation resulting from continuum mechanics of thin objects. However, this formulation asks for second order spatial derivatives so requires the use of complex elements. In this paper, we present a new way of building the interpolation: First, we use the trianular cubic Bézier shell to allow for a good continuity inside and between the elements and second, we build a kinematic mapping to reduce the degrees of freedom of the element from 10 control points with 3 Degrees of Freedom (= 30 DOFs) to only 3 nodes with 6 DOFs (= 18 DOFs). This reduction allows for good computation performance. This new shell model description is also used to map a smooth surface (for the collision detection and response) on a coarse mechanical mesh to account for the complex contacts that take place during surgical procedures. We demonstrate the convergence and the computational efficiency of our approach as well as its use in two different simulation cases: the planning of surgery for congenital heart disease correction and a preliminary simulation of childbirth.

Categories and Subject Descriptors (according to ACM CCS): I.3.5 [Computer Graphics]: Computational Geometry and Object Modeling—Physically based modeling.

1. Introduction

Thin objects are very frequent in our every-day life. e.g. textiles, paper, leaves etc. Such objects are also very common in anatomical or pathological structures for example tubular structures (like blood vessels, colon ...), thin bag structures (Glisson capsule, aneurysms, ...), and also many others (eyes, skin, ...). In the field of surgical simulation, we try to reproduce or anticipate the mechanical behavior of these structures during the surgery. Thus, it is very important to capture accurately their physical behavior. However, in the same time, it is also very important to maintain fast computation rates, in order to be compatible with the targeted applications (interactive simulation for education and for planning).

This problem has been faced by the computer graphics community but for different applications. For instance, many related works address the problem of the simulation of cloth using for example mass-spring models or bending models [**?**, **?**]. However, these models are discrete so their behavior depends on the mesh and their parameters are not easily measurable. This is completely acceptable for some computer graphics application, where the parameters can be tuned manually to obtain a realistic animation. In the case of medical simulation, more physically based modeling is needed.

Methods based on continuum mechanics and namely theory of elasticity gained on popularity in real-time simulations ever since Terzopoulos [**?**] presented his work on elastic deformation modeling. While the methods using finite elements and theory of elasticity are fairly popular in volumetric deformation modeling [**?**,**?**,**?**], they are still frowned upon in the area of thin structures. Yet, a special field of continuum mechanics has studied the deformations of these objects, and has proposed models based on shell theory. This theory is integrated numerically using Finite Elements [**?**], but there are two issues for obtaining interactive frame rates: (i) the continuum equations require second spatial derivatives that can not be handled with linear interpolation and (ii) the final system of equations that describes the mechanics is non-linear if large deformations are present.

The usual argument against non-linear FEM based methods is that they are computationally too expensive. This is in-

deed true if lots of elements are used to discretize the object. A classical strategy, is to use high-polygonal meshes for rendering or collision detection and response, while maintaining a reasonably low number of elements for the mechanics. The work of Bouthors et al. [**?**] is a good application example of this strategy for surface deformations. But this strategy is not sufficient in our case because the problem of performance is also greatly impacted when using elements with interpolation based on high order polynomials. This type of interpolation is necessary because bending continuum model (based on Kirchhoff-Love theory for thin plates) requires the computation of second spatial derivatives. On the other hand, elements common in mechanics (e.g. Discrete Kirchhoff Triangles) are based on shape functions that are impractical for real-time simulations or don't have explicitly defined shape functions at all. In [**?**], Comas et al. use a hybrid interpolation: linear interpolation is used for in-plane deformations and cubic polynomials for bending deformations. The element is piloted by 3 nodes on which 6 degrees of freedom are defined. From this point of view, our work is derived from this approach but we address several noted drawbacks: the formulation misses one rotational degree of freedom, the cubic interpolation is not symmetric and produces C^0 discontinuities between elements on edges.

Bézier based shell elements in their general form have been presented for both triangular [**?**] and quadrilateral [**?**] elements. High number of simulated DOFs makes such formulations impractical for simulations. Ubach et al. [**?**] presented a rotation-free 3-node shell element based on Bézier triangles where the element geometry is estimated from all the neighbouring elements, not just those sharing an edge. This greatly reduces the sparsity of the global stiffness matrix and consequently increases the computation time. Also, the lack of rotational degrees of freedom limits the application in surgical simulations.

We start by reviewing the definition of Bézir triangle. Then the construction of initial control mesh is described and kinematic link between control points is defined. In Section 3 the FE model is derived and the method of computing the corotational frame is described. In Section 4 we specify how to map a high resolution surface mesh onto the mechanical mesh and give formulas for computing velocities of points on the surface and for projection of forces. Section 5 presents test of convergence, speed analysis and two created applications that use the described shell model. We conclude with discussion of present limitations.

1.1. Contribution

Extending the work of Comas [**?**] we have designed a shell element that uses Bézier interpolation polynomials. Our element solves the problems of the element presented by Comas: (i) it uses all 6 degrees of freedom, (ii) the deformations are completely symmetric and (iii) the boundary between elements is continuous. We present a two stage interpolation: (i) first a kinematic link between 6 DOF nodes of element and nodes of B ézier control mesh is defined and (ii) interpolation function of cubic Bézier triangle is applied to interpolate on the surface of the element. Finally from the kinematic link we derive equations for (i) interpolating velocity at arbitrary position on surface and for (ii) distribution of forces applied on the element surface back to 6 DOF nodes of the element. This allows us to use smooth mapped surface for collision detection and allows the propagation of proper collision response back onto the mechanical mesh.

2. Element Kinematics

In this subsection we propose an element featuring interpolation with high degree of flexibility while still maintaining low number of degrees of freedom. We introduce a new way of building the interpolation by using two stages

1. A Bézier interpolation with cubic polynomial functions is defined on a triangle.
2. A kinematic relation between nodes (6DOFs) defined at the vertices and the control points of the Bézier triangle.

2.1. Bézier Triangle Interpolation

In the rest of the document we define the surface over the triangle:

$$(\xi_1,\xi_2) \in \triangle = \{(\xi_1,\xi_2) \mid \xi_1,\xi_2 \geq 0 \text{ and } \xi_1+\xi_2 \leq 1\} \quad (1)$$

As with other Bézier or B-spline surfaces the Bézier triangle is defined by a mesh of control points that do not necessarily lie on the surface. The general n-th order Bézier triangle requires $(n+1)(n+2)/2$ control points and it's surface is defined as:

$$T(\xi_1,\xi_2) = \sum_{0\leq i+j\leq n} B^n_{i,j}(\xi_1,\xi_2)P_{i,j} \quad (2)$$

where $P_{i,j}$ are the control points and $B^n_{i,j}$ are the bivariate Bernstein basis functions defined as:

$$B^n_{i,j}(\xi_1,\xi_2) = \binom{n}{i\ j}\xi_1^i\xi_2^j(1-\xi_1-\xi_2)^{n-i-j}, \quad (3)$$

$$\binom{n}{i\ j} = \frac{n!}{i!j!(n-i-j)!} \quad (4)$$

where $0 \leq i+j \leq n$.

In our work we use cubic Bézier triangle ($n=3$). Triangle of smaller order would fail to describe all possible deformations of the element and triangles of higher order cannot be described unambiguously with only 18 DOFs available to us without adding further constraints on the control mesh. Cubic triangle is described by 10 control points and the surface

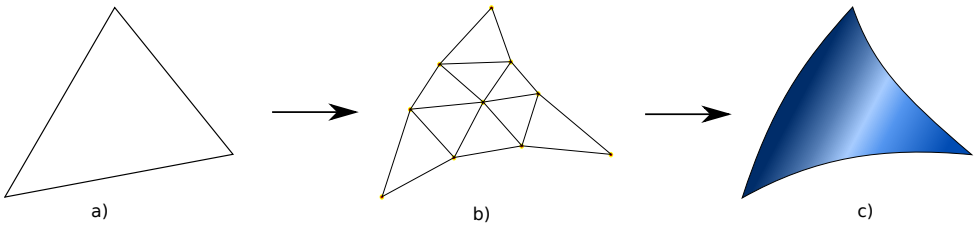

Figure 1: *Two stages of interpolation: First, from triangular with 6DOF nodes (a) a mesh of control points (b) is computed. Then based on definition of the Bézier triangle a surface (c) is interpolated.*

is explicitly defined as (see fig. 2):

$$\begin{aligned} T(\xi_1,\xi_2) = {} & \xi_1^3\,\mathbf{p}_1 + \xi_2^3\,\mathbf{p}_2 + \xi_3^3\,\mathbf{p}_3 \\ & + 3\,\xi_1^2\xi_2\,\mathbf{p}_4 + 3\,\xi_1^2\xi_3\,\mathbf{p}_5 + 3\,\xi_2^2\xi_3\,\mathbf{p}_6 \\ & + 3\,\xi_1\xi_2^2\,\mathbf{p}_7 + 3\,\xi_1\xi_3^2\,\mathbf{p}_8 + 3\,\xi_2\xi_3^2\,\mathbf{p}_9 \\ & + 6\,\xi_1\xi_2\xi_3\,\mathbf{p}_{10} \end{aligned} \tag{5}$$

where $\xi_3 = 1-\xi_1-\xi_2$. By naming n_i the respective values of the Bernstein basis functions linked to control point $\mathbf{p}_i$ we can shortly express (5) as:

$$T = \sum_{i=1}^{10} n_i \mathbf{p}_i \tag{6}$$

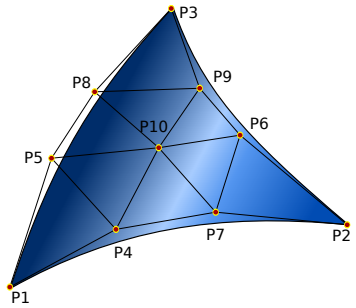

Figure 2: *Cubic Bézier triangle with mesh of 10 control points.*

2.2. Initial Construction of Control Points

Since the simulated object is usually described by a triangular mesh, a procedure for construction of initial Bźier mesh is necessary. Note that special care has to be taken to maintain the continuity between elements across the nodes and edges, we partially employ the method described in [**?**] to maintain C^0 continuity. Each of the control points on the edge is computed as the intersection of:

1. The plane perpendicular to the normal at the vertex.
2. The plane that contains the curve of triangle's contour. The choice is arbitrary, but necessary to maintain C^0 continuity. We choose the plane defined by the edge of the flat triangle and average of the two normals at vertices of the edge.
3. The plane perpendicular to the edge of the flat triangle placed at $1/3$ of the edge length.

C^1 continuity between elements on the edges can also be maintained if special care is taken when computing the position of the central point (see [**?**] or refer to the work of Farin [**?**] for continuity conditions). For simplicity we keep the edge boundary only C^0 and we compute the central point as a function of other 9 points:

$$\mathbf{p}_{10} = \frac{1}{3}\left(\sum_{i=4}^{9}\mathbf{p}_i - \sum_{i=1}^{3}\mathbf{p}_i\right) \tag{7}$$

That way if the element is not flat the central point is slightly elevated and not in the plane of other points thus keeping the curvature of the element.

2.3. Kinematic Between Nodes and Control Points

With 10 control points and 3 DOFs per point it is a total of 30 DOFs for element. We propose a reduction of the number of DOFs by using 3 nodes with 6 degrees of freedom and a kinematic link between control points. Each of the edge control points is attached to nearest corner node. See Figure 3 for the correspondence between edge points and associated corner nodes. In the following text the index $j \in [4;10]$ refers to internal points and index $i \in [1;3]$ to the associated corner point/node if not stated otherwise.

In the initialization phase we remember the position of edge point j in the frame of corner point i:

$$\overline{\mathbf{p}_{i-j}} = R(\theta_i^0)^T \mathbf{p}_{i-j}^0 \tag{8}$$

where $\mathbf{x}_i$ and θ_i are the position and angular position of the node i, $R(\theta_i^0)$ is the rotation matrix for the angular position of node i at the beginning of the simulation and $\mathbf{p}_{i-j}^0 = \mathbf{p}_j^0 - \mathbf{x}_i^0 = \mathbf{p}_j^0 - \mathbf{p}_i^0$ the attachment segment. The 0 in superscript denotes value in rest state.

In each step of the simulation we apply the rigid transformation of frame i on control point j:

$$\mathbf{p}_j = \mathbf{x}_i + \mathbf{p}_{i-j} = \mathbf{p}_i + R(\theta_i)\overline{\mathbf{p}_{i-j}} \tag{9}$$

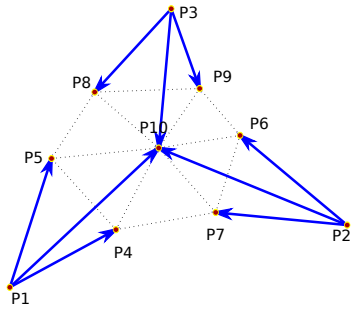

Figure 3: *Internal nodes of control mesh are connected to the corner nodes by kinematic link (shown in blue).*

Similarly for the central point we apply rigid transformation of all frames of the triangle and compute the mean:

$$\mathbf{p}_{10} = \frac{1}{3}\sum_{i=1}^{3}\left(\mathbf{x}_i + R(\theta_i)\overline{\mathbf{p}_{i-10}}\right) \tag{10}$$

Substituting previous equations into (6) we get:

$$T = \sum_{i=1}^{10} n_i \mathbf{p}_i = \sum_{i=1}^{3} n_i \mathbf{x}_i + \sum_{j=4}^{9} n_j \left(\mathbf{x}_i + R(\theta_i)\overline{\mathbf{p}_{i-j}}\right) + \\ + \frac{n_{10}}{3}\sum_{i=1}^{3}\left(\mathbf{x}_i + R(\theta_i)\overline{\mathbf{p}_{i-10}}\right) \tag{11}$$

2.4. Deflection Function

A small variation $\delta\mathbf{x}_i$ of the node i position and $\delta\theta_i$ for the rotation produces the following small variation of position corner points $\delta\mathbf{p}_i$ and edge points $\delta\mathbf{p}_j$:

$$\delta\mathbf{p}_i = \delta\mathbf{x}_i \tag{12}$$

$$\delta\mathbf{p}_j = \delta\mathbf{x}_i + \mathbf{p}_{i-j} \times \delta\theta_i \tag{13}$$

and variation of central point $\delta\mathbf{p}_{10}$:

$$\delta\mathbf{p}_{10} = \frac{1}{3}\sum_{i=1}^{3}\left(\delta\mathbf{x}_i + \mathbf{p}_{i-10} \times \delta\theta_i\right) \tag{14}$$

Thus, finally, it creates a variation of the point position on the surface δT:

$$\delta T = \sum_{i=1}^{10} n_i \delta\mathbf{p}_i \tag{15}$$

or:

Substituting (12)-(14) into (15) one can evaluate the equation in terms of matrix multiplication:

$$\delta T = \mathbf{J}_t \left[\delta\mathbf{x}_1^x, \ldots, \delta\mathbf{x}_3^z, \delta\theta_1^x, \ldots, \delta\theta_3^z\right]^T \tag{16}$$

3. Finite Element Formulation

In the previous section we present an interpolation based on cubic Bézier triangles and a kinematic link between the control points and the element 6DOF nodes. In this section we show how this interpolation can be used to compute a finite element formulation of the shell equations. The model relies on large transformation formulation: the geometrical non-linearities are handled using a corotational formulation, whereas, in the frame of the element, infinitesimal strain theory is used.

3.1. Computation of the Strain Tensor

We define a local system of coordinates (x, y, z) in which the axis z is oriented along the thickness of the element (the computation of the element frame is covered in section 3.4). Given the initial position of the triangle points in this system of coordinate, $\{x_1^0, y_1^0\}$, $\{x_2^0, y_2^0\}$, and $\{x_3^0, y_3^0\}$, we can compute a linear relation between $\{\xi_1, \xi_2, \xi_3\}$ and $\{x, y\}$:

$$\begin{bmatrix} \xi_1(x,y) \\ \xi_2(x,y) \\ \xi_3(x,y) \end{bmatrix} = \begin{bmatrix} 1 & 1 & 1 \\ x_1^0 & x_2^0 & x_3^0 \\ y_1^0 & y_2^0 & y_3^0 \end{bmatrix}^{-1} \begin{bmatrix} 1 \\ x \\ y \end{bmatrix} \tag{17}$$

Thus, we can locally compute some spatial derivatives, along x and y on the interpolation function vector T (which has three components (T^x, T^y, T^z)) . The 2D tensor F of the in-plane deformations can be computed as a 2×2 matrix where the line k and the column l alternatively represent x and y:

$$F_{kl} = \frac{\partial T^k}{\partial l} = \sum_{i=1}^{3} \frac{\partial T^k}{\partial \xi_i}\frac{\partial \xi_i}{\partial l}; \quad k \in \{x,y\}, l \in \{x,y\} \tag{18}$$

The term $\frac{\partial T^k}{\partial \xi_i}$ can be derived from the expression of $T(\xi_1, \xi_2, \xi_3)$ in equation (11) and $\frac{\partial \xi_i}{\partial l}$ from equation (17). The in-plane displacement gradient can be obtained with tensor F:

$$\frac{\partial u^k}{\partial l} = F_{kl} - I_{kl} \tag{19}$$

where $\mathbf{I}$ is the identity matrix ($I_{kl} = \{1 \text{ if } l = k\,;\, 0 \text{ if } l \neq k\}$). The displacement gradient can be linked to the degrees of freedom of the shell using the expression of the displacement of a point δT (15) on the shell surface:

$$\nabla u = \nabla(\delta T) \tag{20}$$

In the local frame of the element, we rely on infinitesimal strain theory, thus the expression of the strain tensor (using Voigt notations) is:

$$\begin{bmatrix} \varepsilon_{xx} \\ \varepsilon_{yy} \\ 2\varepsilon_{yx} \end{bmatrix} = \begin{bmatrix} \frac{\partial u^x}{\partial x} \\ \frac{\partial u^y}{\partial y} \\ \frac{\partial u^x}{\partial y} + \frac{\partial u^y}{\partial x} \end{bmatrix} \tag{21}$$

For the bending formulation, the shell theory uses a measure of the out of plane-displacement $u^z(x,y)$ The strain tensor is based on Kirchhoff-Love theory for thin plates:

$$\begin{bmatrix} \varepsilon_{xx} \\ \varepsilon_{yy} \\ 2\varepsilon_{yx} \end{bmatrix} = \begin{bmatrix} -z\frac{\partial^2 u^z}{\partial x^2} \\ -z\frac{\partial^2 u^z}{\partial y^2} \\ -2z\frac{\partial^2 u^z}{\partial xy} \end{bmatrix} \tag{22}$$

where z is the local coordinate along z-axis of the point for which the strain is measured.

As the term $\frac{\partial \xi_i}{\partial l}$ provides a constant value, the second derivatives can be computed quite easily, for instance, for ε_{xx}:

$$\frac{\partial^2 u^z}{\partial x^2} = \sum_{i=1}^{3} \frac{\partial}{\partial \xi_i} \left(\sum_{j=1}^{3} \frac{\partial T^z}{\partial \xi_j} \frac{\partial \xi_j}{\partial x} \right) \frac{\partial \xi_i}{\partial x} = \sum_{i=1}^{3} \sum_{j=1}^{3} \frac{\partial^2 T^z}{\partial \xi_i \xi_j} \frac{\partial \xi_j}{\partial x} \frac{\partial \xi_i}{\partial x} \tag{23}$$

3.2. Shell Element

As previously described, the shell element's strain is a combination of two types of deformations, as shown in figure 4. We can distribute these two deformations into two groups of degrees of freedom at the nodes' level.

- Elastic membrane defining deformations in plane of the element. It encompasses deformations like stretching and shearing. It corresponds to the in-plane displacements u_x, u_y. At the nodes' level, it corresponds to in-plane displacements $\delta\mathbf{x}_x, \delta\mathbf{x}_y$ and rotation θ_z.
- Bending plate defining deformation due to out of plane bending displacement u_z. It is influenced by 3 degrees of freedom: rotation around two axes θ_x, θ_z and out of plane displacement $\delta\mathbf{x}_z$.

Combining both elements we make use of all 6 degrees of freedom available at each node of the element.

3.3. Stiffness Matrix

In order to compute FEM-based stiffness for the shell element, we need to compute the strain-displacement matrix between strains and nodes' displacements. As we have separated the formulations for the in-plane and the out of plane displacements, we obtain two strain-displacement matrices (of size 3×9).

$$\begin{bmatrix} \varepsilon_{xx} \\ \varepsilon_{yy} \\ 2\varepsilon_{yx} \end{bmatrix} = \mathbf{J}_m \begin{bmatrix} \delta\mathbf{x}_1^x \\ \delta\mathbf{x}_1^y \\ \delta\theta_1^z \\ \vdots \end{bmatrix} \tag{24}$$

where $\mathbf{J}_m$ maps the membrane in-plane displacements and

$$\begin{bmatrix} \varepsilon_{xx} \\ \varepsilon_{yy} \\ 2\varepsilon_{yx} \end{bmatrix} = \mathbf{J}_b \begin{bmatrix} \delta\mathbf{x}_1^z \\ \delta\theta_1^x \\ \delta\theta_1^y \\ \vdots \end{bmatrix} \tag{25}$$

where $\mathbf{J}_b$ maps the bending displacements.

To compute the values in matrices $\mathbf{J}_m$ and $\mathbf{J}_b$, we can reuse equation (15) and apply the spatial derivatives. To illustrate, let's take one value of the matrix $\mathbf{J}_m$ (first line, fourth column), we first take the kinematics between δT_x and the concerned degree of freedom $\delta\mathbf{x}_2^x$ from equation (15):

$$\frac{\delta T_x}{\delta \mathbf{x}_2^x} = (n_2 + n_6 + n_7 + \frac{n_{10}}{3}) \tag{26}$$

Then, we apply the spatial derivatives to compute the variation of ε_{xx} due to a variation of $\mathbf{x}_{2x}$:

$$\delta\varepsilon_{xx} + = \underbrace{\left(\sum_{i=1}^{3} (\frac{\partial n_2}{\partial \xi_i} + \frac{\partial n_6}{\partial \xi_i} + \frac{\partial n_7}{\partial \xi_i} + \frac{1}{3}\frac{\partial n_{10}}{\partial \xi_i}) \frac{\partial \xi_i}{\partial x} \right)}_{J_{m14}} \delta\mathbf{x}_2^x \tag{27}$$

Assuming constant thickness t of the element and integrating over the volume of the element we compute the stiffness matrices for the elastic membrane and bending plate respectively:

$$\mathbf{K}_m = \iiint_V \mathbf{J}_m^T \mathbf{M} \mathbf{J}_m \, dV \tag{28}$$

$$\mathbf{K}_b = \iiint_V \mathbf{J}_b^T \mathbf{M} \mathbf{J}_b \, dV \tag{29}$$

where $\mathbf{M}$ is the material matrix. To keep the system simple we use linear Hooke's law for isotropic materials.

Because the deflection field for the element is non-linear (15) in position the integrals (28) and (29) have to be computed using numerical integration. In our implementation we employ 6-point Gaussian quadrature for integration over triangle area.

3.4. Corotational Formulation

It is known that the Cauchy's strain tensor (21) is not rotation invariant [**?**, **?**] and produces ghost forces for rigid rotations.To keep the system linear and to deal with rigid body rotations we compute the displacements and forces in corotational frame. At each step of the simulation we compute the local frame for every element, the resulting internal force of deformed element is then computed relative to this frame:

$$F_e = \mathbf{R}_e^T (\mathbf{K}_e u) = \mathbf{R}_e^T \left(\mathbf{K}_e (\mathbf{R}_e x - \mathbf{R}_e^0 x^0) \right) \tag{30}$$

where $\mathbf{R}$ and $\mathbf{R}_0$ are rotation matrices of the frame for current and rest mesh respectively.

The corotational frame is computed using following steps:

- One axis is aligned to one edge of the element: $\mathbf{e}_1 = \mathbf{x}_2 - \mathbf{x}_1$
- A normal to the plane of the element is computed: $\mathbf{e}_3 = \mathbf{e}_1 \times (\mathbf{x}_3 - \mathbf{x}_1)$

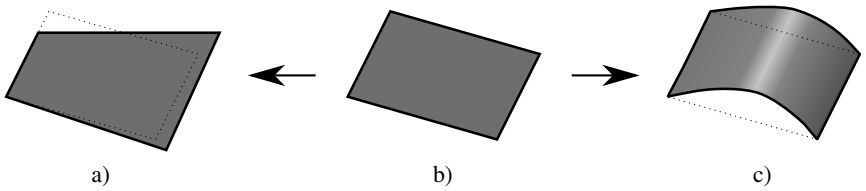

Figure 4: *Plate deformations: a) membrane deformations like stretching or shearing, due to in-plane displacements b) undeformed; c) bending deformation, due to out of plane displacements*

- Second axis in the plane of the element is computed: $\mathbf{e}_2 = \mathbf{e}_1 \times \mathbf{e}_3$

Correct choice of the corotational frame is a tricky problem [**?**] and incorrect frame for the membrane element can have adverse effects on the simulation. This choice of the frame where we align one axis with one of the edges of the triangle is not a perfect one and we have experienced non-negligible difference in deformation of equilateral triangle depending on whether an axis was aligned to the edge favored by the deformation or to one of the two other edges.

A polar decomposition on the deformation gradient $F = \nabla u + \mathbf{I}$ can be used to extract a rotation component from the deformation of the element [**?**,**?**]. Because we are only fixing the membrane element we can perform the polar decomposition in 2D with the following formula:

$$\mathbf{R} = \mathbf{F} + \text{sgn}(\det(\mathbf{F})) \begin{pmatrix} F_{22} & -F_{21} \\ F_{12} & F_{11} \end{pmatrix} \tag{31}$$

and subsequently normalizing the columns. We then rotate the corotational frame in opposite direction. The deformation gradient depends on frame which means multiple iterations of polar decomposition can be performed. Even single iteration improves the final frame but more are usually necessary (up to 20) which severely degrades the performance. Polar decomposition tends to "overshoot" the ideal rotation angle and each subsequent step performs a rotation in the direction opposite to the rotation in previous step. By properly scaling the rotation angle we are able to minimize the amount of needed iterations down to 5. The best value for the scaling factor depends on the simulated problem, but we have experimentally localized the best value to be somewhere between 0.6 and 0.65. We choose to use the value 0.61.

We employ two stop conditions for the iteration process: (i) the change in rotation angle is less than 10^{-6}, and (ii) the maximum number of iterations is set to 5.

If the first condition is not met in 5 iterations we still have very good approximation of the ideal frame. In some situations the process is not able to reach the ideal frame quickly because either the convergence for the element is too slow or the process is oscilating around a certain configuration. In either case performing more iterations is unnecessary.

4. Application of High Resolution Mesh

Because of the bending property of the shells relatively few elements are necessary to simulate curved surface. To enrich the visual experience from the simulated object it is desirable to use more triangles in the areas with high curvature during rendering. Besides the visual accuracy the high resolution mesh correctly modeling the curved areas is needed for interaction and correct simulation of collisions and constraints. To handle these, we also need to know how to map the velocities from the mechanical mesh onto the high resolution mesh and how to map forces acting on the high resolution mesh back onto the mechanical mesh. This high resolution mesh may be obtained simply by refining the triangles of mechanical mesh.

4.1. Surface Mesh

To project vertices onto the surface of shells we use the fact that the geometry of is based on the formulation of cubic Bézier triangle (5). For every vertex of the high resolution mesh we first find the corresponding triangle on the mechanical mesh that is closest to the vertex and assign barycentric coordinates on the triangle to this vertex. After every step of the simulation points of the control mesh are updated based on the kinematic link described in section 2 and the high resolution mesh is updated using the assigned barycentric coordinates and the function of the surface (5). An alternative approach is to subdivide the Bézier triangles as necessary.

4.2. Nodal Velocities of the Surface Mesh

For contact modeling it is necessary to know the velocity of the contact points. The values are directly available to us for the nodes of the mechanical mesh but they can also be computed for any point on the surface. Differentiating (6) by time we get:

$$\dot{T} = \sum_{i=1}^{10} n_i \dot{\mathbf{p}}_i = \sum_{i=1}^{10} n_i \mathbf{v}_i \tag{32}$$

For the corner control points we already have the velocities $\mathbf{v}_i$. For the internal control points we need to compute them. We again make use of the kinematic link between control points and control points. For the edge points we applied

to motion of a rigid body:

$$\mathbf{v}_j = \mathbf{v}_i + \omega_i \times \mathbf{p}_{i-j} \tag{33}$$

where i is the index of the corner node the edge node is attached to and ω_i is the angular velocity at the node. For the central node the formula is based on the derivate of equation (10):

$$\mathbf{v}_{10} = \frac{1}{3}\sum_{i=1}^{3}(\mathbf{v}_i + \omega_i \times \mathbf{p}_{i-10}) \tag{34}$$

Again after substitution of (33) and (34) into (32) one can evaluate $\dot{T}$ as $\dot{T} = \mathbf{J}_t \left[\mathbf{v}_1^x, \ldots, \mathbf{v}_3^z, \omega_1^x, \ldots, \omega_3^z\right]^T$.

4.3. Projection of Surface Forces

For physical interaction with the object one also requires that the forces applied on the object's surface are transmitted back as forces and torques on the vertices of the mechanical mesh. The influence of force $\mathbf{f}$ acting on the fine mesh is transfered using the associated barycentric coordinates through the control points by the formula:

$$\mathbf{f}_1 = \sum_{i\in\{1,4,5\}} n_i\mathbf{f} + \frac{n_{10}}{3}\mathbf{f} \tag{35}$$

$$\mathbf{f}_2 = \sum_{i\in\{2,6,7\}} n_i\mathbf{f} + \frac{n_{10}}{3}\mathbf{f} \tag{36}$$

$$\mathbf{f}_3 = \sum_{i\in\{3,8,9\}} n_i\mathbf{f} + \frac{n_{10}}{3}\mathbf{f} \tag{37}$$

Similarly we can compute the torques applied through edge control points and the central control points:

$$\tau_1 = \mathbf{p}_{1-4}\times(n_4\mathbf{f}) + \mathbf{p}_{1-5}\times(n_5\mathbf{f}) + \frac{1}{3}\mathbf{p}_{1-10}\times(n_{10}\mathbf{f}) \tag{38}$$

$$\tau_2 = \mathbf{p}_{2-6}\times(n_6\mathbf{f}) + \mathbf{p}_{2-7}\times(n_7\mathbf{f}) + \frac{1}{3}\mathbf{p}_{2-10}\times(n_{10}\mathbf{f}) \tag{39}$$

$$\tau_3 = \mathbf{p}_{3-8}\times(n_8\mathbf{f}) + \mathbf{p}_{3-9}\times(n_9\mathbf{f}) + \frac{1}{3}\mathbf{p}_{3-10}\times(n_{10}\mathbf{f}) \tag{40}$$

Or again as $\left[\mathbf{f}_1^x, \ldots, \mathbf{f}_3^z, \tau_1^x, \ldots, \tau_3^z\right]^T = \mathbf{J}_i^T\mathbf{f}$. Note that these formula fulfill the physical principle of virtual power: $P = \dot{T}\cdot f = \sum_{i=1}^{3} V_i \cdot f_i + \omega_i \cdot \tau_i$

5. Validation and Results

In subsections 5.1 and 5.2 we present tests to validate the convergence of our Bézier shell element. As a reference we use a combination of two elements that are very well established in the area of mechanical modeling. Our results are compared with the shell element composed of Discrete Kirchoff Triangle (DKT) element [**?**] for bending plate and optimal ANDES element [**?**] for membrane. For comparison the results for the model of O. Comas [**?**] are also presented. Implicit Euler integration scheme was used to solve the system. This allows us to use large time steps needed for real-time performance of at least 25 Hz. Subsection 5.3 presents evaluation of the computational complexity of our solution. Last subsections are dedicated to presentation of application of our model.

5.1. Roof Test

To validate the results of the element we have performed a modified version of a test known as Scordelis-Lo roof. It simulates a cylindrical roof under self-weight. The geometry is defined by 80° cylindrical patch of length $L = 50$, radius $r = 25$ and thickness $t = 0.25$ and is discretized into mesh of NxN vertices. The physical parameters are $E = 4.32 \times 10^8$ and $\nu = 0$. The roof is loaded with uniformly distributed load $q = 90$ per unit area. The curved edges are clamped at both ends to avoid rigid body movement. This is the only difference from the original Scordelis-Lo roof test where the edges are free to move in longitudinal direction. The Figure 5 shows the vertical displacement of midpoint on the free edge.

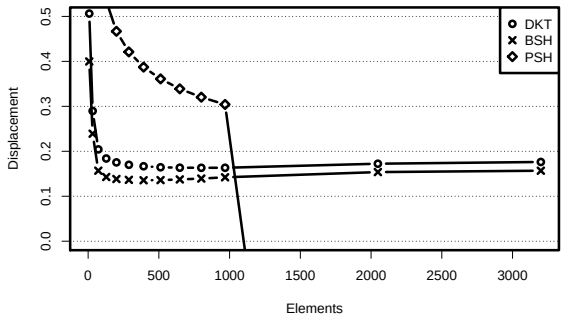

Figure 5: *Convergence for modified Scordelis-Lo roof test of DKT+ANDES element (DKT), our Bézier shell element (BSH) and element of O. Comas (PSH) with polynomial shape function.*

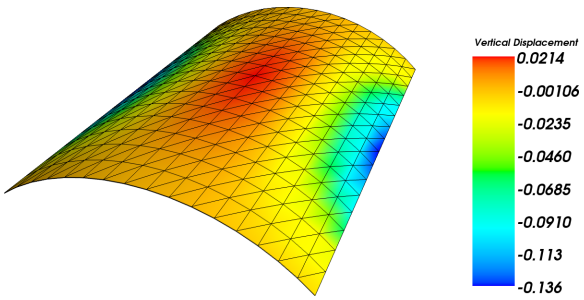

Figure 6: *Coloured plot of vertical displacement in the roof test*

5.2. Hemisphere Test

Second performed test is a hemisphere with hole subjected to two opposing forces at the base. The geometry is described by a hemisphere with radius $r = 10$ and thickness $t = 0.04$ with 18° hole at the top and the geometry is discretized into the grid of NxN vertices per quadrant. The physical parameters are $E = 6.825 \times 10^7$ and $\nu = 0.3$. The hemisphere is at it's base subjected to two opposing outwards forces and two opposing inwards forces with magnitude $P = 4$. To avoid rigid body movement we have constrained two opposite nodes at the top of the hemisphere. The radial displacement for one point with applied load is shown in Figure 7.

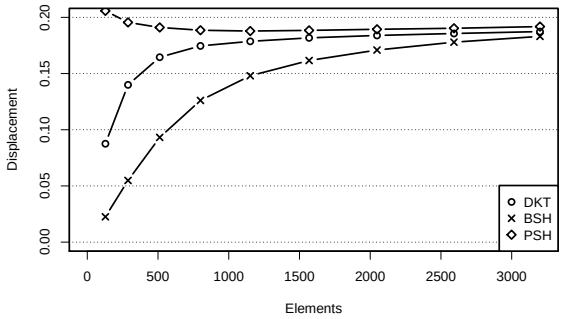

Figure 7: *Convergence for test on hemisphere with 18° hole for the DKT+ANDES element (DKT) and our Bézier shell element (BSH) and element of O. Comas (PSH) with polynomial shape function.*

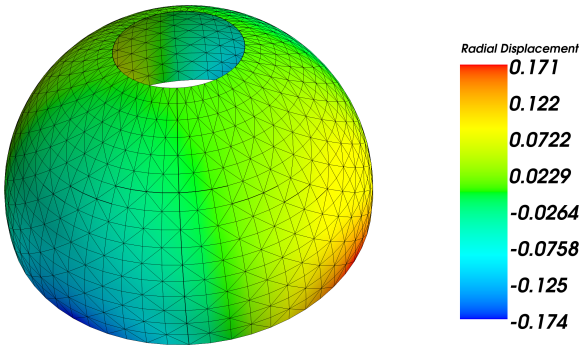

Figure 8: *Coloured plot of radial displacement in the hemisphere test*

5.3. Computation Speed

In figure 9 we present performance of the element in terms of frames per second. All tests were performed on machine equipped with Dual-Core AMD Opteron Processor 2218 and 3 GB of RAM. We used a conjugate gradient solver to solve the system. Our implementation is without optimizations and only single CPU core was used for the simulation. The values reported are for the raw physical simulation with (B) and without (A) the frame fixing method described in subsection 3.4. We also present results including the mapping of high-resolution mesh onto the mechanical mesh for visualisation purposes or collisions.

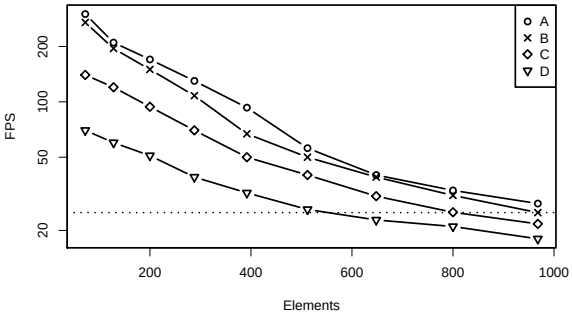

Figure 9: *Performance in frames per seconds for different number of elements. Shows values for element with frame fixing (B) method described in subsection 3.4 and without it (A), and with mesh of 3 200 (C) or 10 082 (D) triangles mapped onto the mechanical mesh.*

We can see that the frame fixing technique is really fast and we are able to simulate up to 1 000 elements and still maintain visual refresh rate of 25 FPS. While our implementation was only single-threaded and the mechanical mapping wasn't off-loaded to another CPU core we were still able to provide good results for more than 600 elements mesh with 10k triangles.

Both DKT+ANDES and our element are purely linear which makes their raw simulation time comparable. The hybrid PSH element require evaluation of coefficients of the polynomial during simulation which makes it inferior in performance.

5.4. Planning of Congenital Heart Disease Correction Surgery

Surgical interventions in infants with congenitally malformed great arteries and hearts are extremely challenging due to the complex and heterogeneous nature of their disease patterns. At present, cardiac surgeons rely on non-invasive imaging for patient-specific examination and a set of preoperative sketches with varying approaches to the surgical procedure. However, the most promising approach is often chosen during the actual open-heart surgery, when surgeons get a more concrete idea of possible outcomes. The important decisions have to be made in very short time and are strongly dependent on the surgeon's experience.

To be able to predict results of complex surgical procedures a surgical simulation system can be employed. We have developed a prototype [**?**] of new surgical simulation

system for preoperative planning. The element has been used to model deformations of blood vessels and artificial patches. By performing a set of topological changes we simulate the process of incising the vessel and suturing on a prespecified boundary. By iteratively adjusting the underlaying mesh we search for the elastic equilibrium.

5.5. Simulation of Uterus Deformation During Birth

The purpose of this simulator is to reproduce the tissue deformations that are encountered during a delivery. The goal is to evaluate the mechanical stresses which are applied on the anatomical structures (like the uterus but also the ligaments that sustain the uterus).

The data comes from a MRI of a 8 months pregnant woman. The trajectory of the baby-head during the birth was predefined with gynecologists. The model is limited to uterus deformations but will soon account for the vaginal tissues deformation. The collisions response drives the deformations of the uterus. Mechanically, the uterus is modeled with 127 shells that are solved by a direct solver (LDL factorization). Up to 60 instantaneous contact spots need to be mapped on the shell elements. The simulation frame-rate is relatively low (6 FPS) due to contact response computation.

6. Discussion and Limitations

Currently, due to the technical decisions the element has two limitations. It is C^1 continuous at corners and only C^0 on the edges. The lack of continuity may prove to be an issue for example if curvature needs to be computed not only inside the element but also across the corners/edges. In location with high initial curvature more elements may be necessary to properly model the surface.

The second limitation is the choice of rigidly attached control points. While it greatly simplifies the formulation the rigidity may cause issues. In case of large compression of the element the results may be unpredictable. This is however unlikely to occur because the element will bend before reaching such configuration. It may, however, fail to maintain even curvature of the surface if the element is subjected to large stretching. We could investigate more complex kinematic links between nodes and control points and also more continuous surface descriptions.

On the other hand the choices provide a good tradeoff between continuity and simplicity. Only 3×6 DOFs $=$ 18 DOFs are necessary while the Bézier triangle has theoretically 10×3 DOFs $= 30$ DOFs. For complex elements with high number of nodes the interconnection between nodes increases, especially on the vertices where the size of nodes involved depends also on number of neighbouring elements. By keeping the element simple the sparsity of stiffness matrix is not violated.

Our study relies on the assumption that the in-plane and the bending deformation energies can be separated. For complex deformation fields or material constitutive laws this assumption is no more valid. However, we could keep the same way of building the element even with more complex mechanics. For instance the interpolation would allow for derivation of three-dimensional non-linear strain tensors. We will investigate this topic in our future work.

7. Conclusion

In this paper we presents new shell element with interpolation functions based on Bézier triangle which is suitable for real-time simulations. Compared to the elements commonly used in mechanics, the interpolation is less complex and ensures at least a C^0 continuity in the worst case. (Many shell elements are not even C^0 when the shape is not flat).

It fixes problems of the previous model [**?**] based on cubic polynomials, namely: element is based on all 6 DOFs, deformations of the element are symmetric, it is C^1 for the case of initial flat shape and, on the other cases, it is C^0 continuous on the edges and provides C^1 continuity in corners of the element. This is essential for good visual experience if high-resolution mesh is mapped on the mechanical mesh. A method of improving the corotational frame to keep the deformation symmetric is also presented in the paper.

The formulation of Bézier triangle provides a clean way of mapping high-resolution mesh for better visualization. We also provide a technique of computing velocities of any point on the surface and formula for back-projection of forces applied on the surface onto the mechanical mesh. Those are essential components for the surgical simulators that need accurate contact geometry modeling without an overflow of computation. The efficiency of the approach is demonstrated through convergence tests, computation time benchmark and application to two different simulation cases.

In future work, we will investigate the use of more complex constitutive laws and also the use of GPU for parallel and faster compuation.

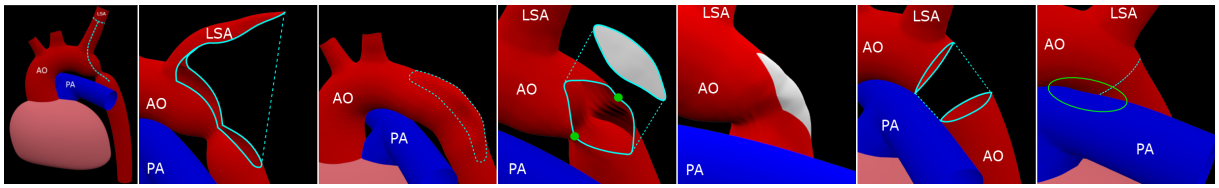

Figure 10: *Screenshots of the simulation system prototype for different surgical procedures used to repair a coarctation of an aortic arch. The screenshots are enriched with overlay delineations.*

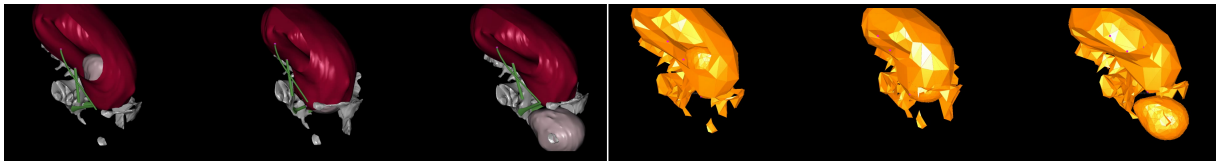

Figure 11: *Screenshots from the simulation of child head during birth. Full visualization (left) and the mechanical mesh (right).*