High Order Elements in Sofa

Hervé Delingette
INRIA Méditerranée, Sophia Antipolis, France

2014

1 High Order Tetrahedral Meshes

1.1 Introduction

We define high order tetrahedral elements in their Bernstein / Bezier form rather than Hermite form. A high order tetrahedral mesh is defined by :

- An underlying tetrahedral mesh $\mathcal{M}$ consisting of a set of *"tetrahedron vertices"* $\mathcal{V}$, edges $\mathcal{E}$, triangles $\mathcal{TR}$ and tetrahedra $\mathcal{T}$. We write V, the number of "tetrahedron vertices", E the number of edges, TR the number of triangles and TE the number of tetrahedra.
- A set of control points $\mathcal{C}$
- A set of quadrivariate Bernstein polynomials allowing to describe the value of a node anywhere on the mesh $\mathcal{M}$

In terms of topology, a high order tetrahedral mesh has more control points than tetrahedron vertices. Below are examples of high order tetrahedral elements of various degree.

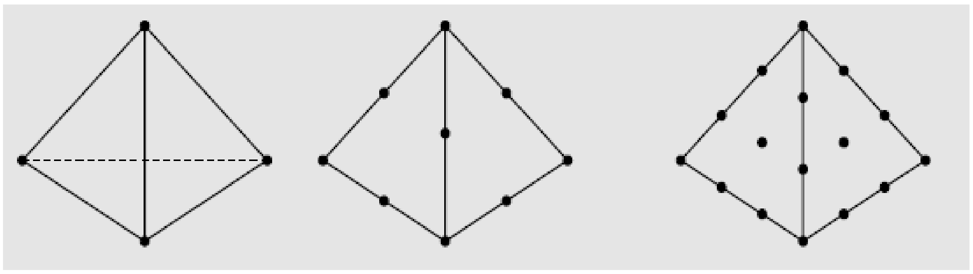

Figure 1: Linear ($d = 1$), Quadratic ($d = 2$) and Cubic ($d = 3$) Tetrahedral Elements

1.1.1 Number of Control Points

If we write $d > 0$ the degree (or order) of a tetrahedral element, then there are :

- V controls points that coincide with the *"tetrahedron vertices"*.
- $(d-1)E$ if $d > 1$ control points that are lying on edges.
- $\frac{(d-1)(d-2)TR}{2}$ if $d > 2$ control points that are lying on triangles.

- $\frac{(d-1)(d-2)(d-3)TE}{6}$ if $d > 3$ control points that are lying on tetrahedra.

Thus the total number of control points are :

$$C = V + (d-1)E + \frac{(d-1)(d-2)TR}{2} + \frac{(d-1)(d-2)(d-3)TE}{6}$$

1.1.2 Tetrahedron Bezier Indices

We use specific notations of control points inside a tetrahedron which we call *Tetrahedron Bezier Indices* (TBI). A TBI $p \in \mathbb{N}^+ \times \mathbb{N}^+ \times \mathbb{N}^+ \times \mathbb{N}^+$ is a 4-plet of positive natural numbers that indicate their relative position in the high order element.

Thus for an element of degree d, $\mathbf{p} = (i, j, k, l)$ is such that $|\mathbf{p}| = i + j + k + l = d$. The four TBI $(d, 0, 0, 0)$, $(0, d, 0, 0)$, $(0, 0, d, 0)$, $(0, 0, 0, d)$ coincides with the 4 tetrahedron vertices while $(0, i, 0, j), i > 0, j > 0$ is lying on the edge linking the second and fourth vertex and $(0, i, j, k), i > 0, j > 0, k > 0$ is lying on the triangle opposite to the first vertex. We write $\mathbf{C_p}$ the control point associated with indices (i, j, k, l).

1.1.3 Tetravariate Bernstein Polynomial

The control points are used to define a parametric volume in space. The parameters are the barycentric coordinates (r, s, t, u) such that $r + s + t + u = 1$ and $0 \le r, s, t, u \le 1$. The shape functions are the tetravariate Bernstein polynomials $B^d_{i,j,k,l}(r, s, t, u)$ of degree d that are themselves parameterized by four indices (i, j, k, l) such that $i + j + k + l = d$ with the following expression:

$$B^d_{i,j,k,l}(r, s, t, u) = \frac{d!}{i!j!k!l!} r^i s^j t^k u^l$$

For given degree d there are $N_d = 4+6*(d-1)+2*(d-1)*(d-2)+(d-1)*(d-2)*(d-3)/6$ such polynomials. To simplify notation, we use the same Tetrahedron Bezier Indices for the Bernstein polynomial as for the control points. Therefore $B^d_{\mathbf{p}}(r, s, t, u) = B^d_{i,j,k,l}(r, s, t, u)$.

With this notation, the position of a point parameterized by (r, s, t, u) on a Bezier Tetrahedron is given by :

$$\mathbf{C}(r, s, t, u) = \sum_{\|\mathbf{p}\|=d} B^d_{\mathbf{p}}(r, s, t, u)\mathbf{C_p}$$

1.2 SOFA Implementation

1.3 Layout of Degrees of Freedom in SOFA

In SOFA, the degrees of Freedom (DOF) are stored into a set of arrays inside objects called MechanicalState. We have chosen to store the $\mathcal{C}$ DOFs of a Bezier Tetrahedral mesh inside a single MechanicalState. Therefore a specific order of the DOFs in a MechanicalState has been defined. Figure 3 shows this ordering of control points. First of all, the control points associated with the tetrahedron vertices are stored, then those associated with edges, triangle and tetrahedra.

There are however some issues. In a tetrahedral mesh, edges and triangles are not oriented (*e.g.* a triangle is common to 2 tetrahedra and is ordered differently among each tetrahedron) and therefore the order in the DoF array may not reflect the order in each tetrahedron. It is the role of the BezierTetrahedronSetTopologyContainer class to provide a proper ordering.

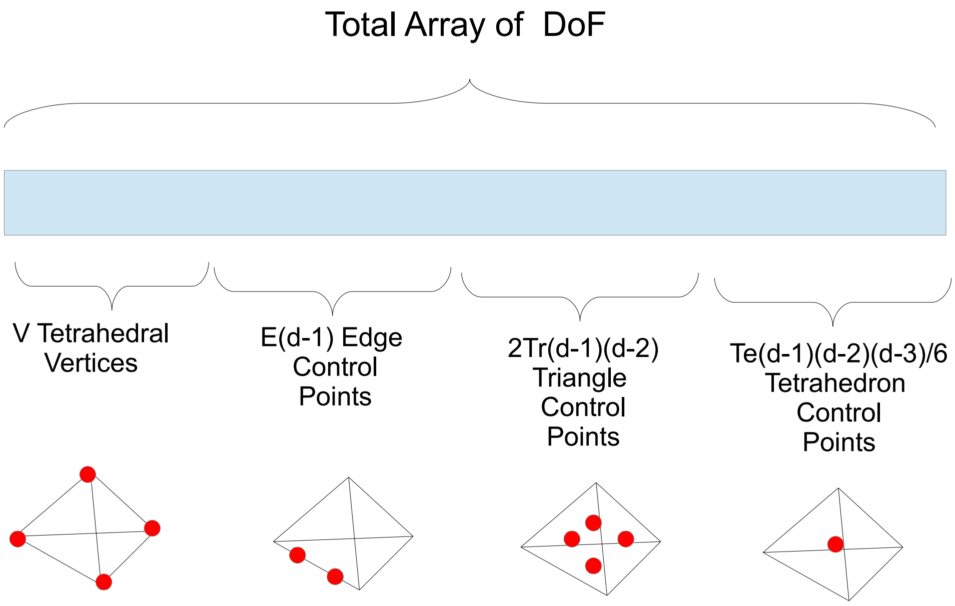
Total Array of DoF
V Tetrahedral Vertices
E(d-1) Edge Control Points
2Tr(d-1)(d-2) Triangle Control Points
Te(d-1)(d-2)(d-3)/6 Tetrahedron Control Points

Figure 2: Layout of Degrees of Freedom of Bezier Tetrahedral meshes inside a MechanicalState object

Second, since there are several control points for a given edge, triangle and tetrahedron, it is important to specify the ordering inside each element. Figure?? show in a simplified way how the control points are numbered for an edge, triangle and tetrahedron. In a nutshell, the element are stored in increasing lexicographic order of its barycentric coordinates. Thus on a quartic Tetrahedron element, the points of barycentric coordinates $(1/4, 3/4), (1/2, 1/2), (3/4, 1/4)$ are stored in this order. However, this order may not the one suitable for a given Bezier tetrahedron as the edge is shared by several tetrahedra.

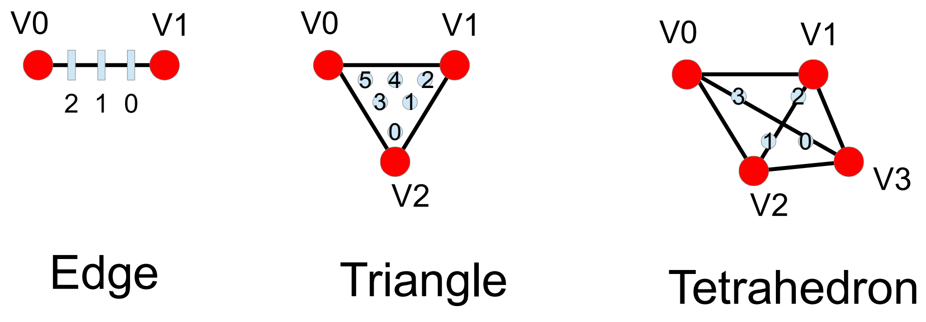

Figure 3: Layout of Degrees of Freedom of Bezier Tetrahedral meshes inside a MechanicalState object

1.4 BezierTetrahedronSetTopologyContainer Class

This class provides an interface to the array of DoFs store in the MechanicalState object. More precisely, a control point can specified in 3 different ways :

- A *global index* which is its rank in the array of size C.
- A tetrahedron index and a Tetrahedron Bezier Index (like $(2,0,2,0)$ on a tetrahedron of order 4).
- A tetrahedron index and its *local index*. The local index is an integer between 0 and N_d which give the rank of the control points in this local (tetrahedron specific) list of control points.

The class thus provides function to move from one representation of a control point to another.

The function *void getGlobalIndexArrayOfBezierPointsInTetrahedron(const size_t tetrahedronIndex, VecPointID & indexArray)* is particularly useful as it outputs an array of N_d global indices of the control points located inside a given tetrahedron. The tetrahedron Bezier indices of those control points are always the same for a given degree d and is given by the array returned by the function *sofa::helper::vector<TetrahedronBezierIndex> getTetrahedronBezierIndexArray() const;*.

Obviously the class can return the degree of Bezier tetrahedra (they are all of the same degree). Note that it provides both the total number C of control points $getNbPoint()$ and the number V of tetrahedral points.

1.5 BezierTetrahedronSetGeometryAlgorithms Class

This class provides basic geometric functions :

- *Coord computeNodalValue(const size_t tetrahedronIndex,const Vec4 barycentricCoordinate);* which computes the position / value of a node given the index of a tetrahedron and its barycentric coordinate.
- *Real computeBernsteinPolynomial(const TetrahedronBezierIndex tbi, const Vec4 barycentricCoordinate);* which computes $B^d_{\mathbf{p}}(r,s,t,u)$.

To optimize things it precomputes the Bernstein coefficients of a given degree. If *drawControlPointsEdges* is set to true then it draws the edges of the control point net.

1.6 Mesh2BezierTopologicalMapping Class

This class provide a way to create a Bezier tetrahedral mesh of any order given a tetrahedral mesh. The control points are simply linearly interpolated from the tetrahedral vertices : a control point of index $\mathbf{p} = (i,j,k,l)$ is computed with the barycentric coordinates $(\frac{i}{d},\frac{j}{d},\frac{k}{d},\frac{l}{d})$. The mapping automatically sets the degree and the number of tetrahedral vertices in the object BezierTetrahedronSetTopologyContainer.

Note that the position of the mapping in the scene has be after BezierTetrahedronSetTopologyContainer (see the example in BezierTetrahedronTopologicalMapping.scn)