Scalar Tagged PN Triangles

Tamy Boubekeur[+] Patrick Reuter[*] Christophe Schlick[+]

[+]: LaBRI INRIA CNRS University of Bordeaux [*]:LIPSI ESTIA

Abstract

*This paper presents a new technique to convert a coarse polygonal geometric model into a smooth surface interpolating the mesh vertices, by improving the principle proposed by Vlachos et al. in their "Curved PN-Triangles". The key idea is to assign to each mesh vertex, a set of **three scalar tags** that act as shape controllers. These scalar tags (called sharpness, bias, and tension) are used to compute a procedural displacement map that enriches the geometry, and a procedural normal map that enriches the shading. The resulting technique offers two majors features: first, it can be applied on meshes of arbitrary topology while always generating surfaces with consistent behaviors across edge and vertex boundaries, second, it only involves operations that are purely local to each polygon, which means that it is very well suited for hardware implementations.*

1. Introduction

Compared to other subdivision schemes [ZS00] or mesh smoothing techniques, *Curved PN-Triangles*, a totally local refinement scheme introduced by Vlachos et al. [VPBM01], is much better suited to hardware implementation, since no adjacency information between triangles has to be stored and managed. So, starting from a coarse mesh defined by the user, an interpolating refined mesh can be generated on-the-fly at rendering time. The most innovative idea of PN-Triangles, compared to previous work, is to relax the constraint of high-order geometric continuity, and to show that a simple *visual smoothness* is sufficient for many applications. This visual smoothness is obtained by computing, simultaneously but independently, a displacement field used to enrich the geometry of each triangle, and a procedural normal field used to enrich its shading. In order to offer a greater control on the initial coarse mesh, this paper proposes to assign to each vertex of this coarse mesh, a set of *three scalar tags* that act as intuitive shape controllers, namely sharpness, tension and bias. The area of influence of these shape controllers is very local but is sufficient to guarantee consistent local surface features, such as curvature values around vertices or tangent plane discontinuities across edges.

2. Description of Scalar Tags

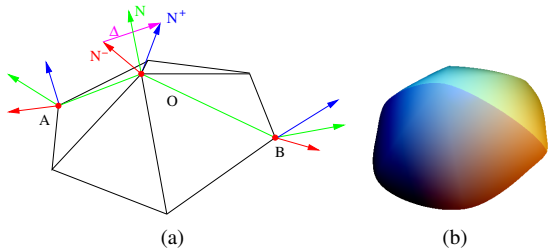
2.1. Local surface analysis

Indexed faces sets have become the most common data structure to store polygonal meshes, as it avoids the duplication of the vertex coordinates. But it has also one major consequence: the only adjacency relationship stored in the data structure are the indices of common vertices shared by two neighboring polygons. Thus the only way to get a consistent behavior of the surface across polygon boundaries is to store the shape parameters on a per vertex basis and to ensure that the influence of all the shape parameters is strongly localized around each vertex.

Unfortunately, if a per-vertex storage is well-adapted to per-vertex shape parameters such as local tangent plane or local curvature, it is much less adapted to per-edge features that may exist in the geometry, such as creases or straight lines. So, to be able to correctly account for per-edge features, we impose some constraints on the *one-ring* neighborhood of each vertex. Let us consider Figure 1: a crease passes through the vertex O and cuts the underlying triangle fan in two sub-fans. An average normal vector N^+ and N^- can be computed for each sub-fan, by simply averaging the normal vectors of the included triangles. The sharp crease is then implicitly defined by the three tagged vertices A, O and B. The corresponding normal discontinuity can be simply encoded, by applying a kind of Haar filtering on the two normal vectors N^+ and N^-: we store the average normal vector $N = N^+ + N^-$ (which is normalized to unit length) and a difference vector $\Delta = N^+ - N^-$. So $\Delta = 0$ corresponds to a smooth vertex. In the remainder of the paper, we will use the word “tagged” to specify a vertex with a non-null Δ and the word “untagged” otherwise. To be able to always keep a local decision about per-edge features, we impose the following restriction on the local configuration:

1. A tagged vertex can have 2 tagged neighbors at most.
2. A triangle can have 2 tagged vertices at most.

The first restriction ensures that only one crease passes through a given vertex. If not, this would mean that we need more than one vector Δ to encode the normal discontinuity



(a) (b)

Figure 1: *(a) At the vertex level, the green sharp crease can be encoded by a simple vector Δ. (b) This additional per-vertex data locally controls the underlaying per-triangle smooth surface generation.*

at this vertex. As an extension, encoding several vectors Δ would allow to represent multiple sharp creases, but at the price of a more important per-vertex data set. The second restriction make unambiguous the difference between two distinct creases that are separated by only one triangle, and a crease that loops around a single triangle. Note that some simple local remeshing step can remove this limitation (split for instance).

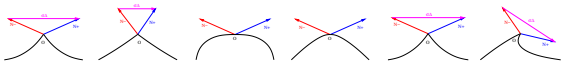
2.2. Shape parameters through scalar tags

Compared to the original PN-Triangle model, the inclusion of the normal discontinuity vector Δ allows us to generate different displacement fields and normal fields on both sides of a crease edge. We propose now to define additional per-vertex scalar values (i.e. *scalar tags*) to offer an even more accurate control of the local geometry. We have selected three shape parameters that are particulary well adapted to the control of sharp creases. We first describe these three shape parameters independently of the underlying refinement technique.

The first scalar tag $\sigma \in [0,\infty)$ is called *sharpness*: it defines the divergence of normal vectors across the two sides of the crease, by interpolating between totally smooth ($\sigma = 0$) and totally sharp ($\sigma = \infty$) configurations (see Figure 2(a)).

The second scalar tag $\theta \in [-1,1]$ is called *tension*: it corresponds to the usual tension parameter that has been defined in the spline literature [BB83, Far01]. It is used to locally control the curvature of all Bézier boundary curves that are starting from on a given tagged vertex (see Figure 2(b)), and allows to interpolate between three different configurations: tensed Bézier ($\theta > 0$), standard Bézier ($\theta = 0$) and relaxed Bézier ($\theta < 0$).

The third scalar tag $\beta \in [-1,1]$ is called *bias*: it also corresponds to the usual bias parameter that has been defined in the spline literature [BB83, Far01]. It is used to locally control the direction of all Bézier boundary curves that are starting from a given tagged vertex (see Figure 2(c)). Here again three different configurations are interpolated: bias toward N^+ ($\beta > 0$), no bias ($\beta = 0$) and bias toward N^- ($\beta < 0$).



(a) Sharpness (b) Tension (c) Bias

Figure 2: *The role of scalar tags. (a) Sharpness σ controls the normal deviation on sharp creases. (b) Tension θ controls the curvature of boundary curves in the vertex neighborhood. (c) Bias β controls the direction of boundary curves in the vertex neighborhood.*

These three scalar tags are used as shape controllers and they completely drive the mesh refinement. Our experiments have shown that these values, defined by the user, are very intuitive and predictable, even for users not familiar with geometric modeling softwares.

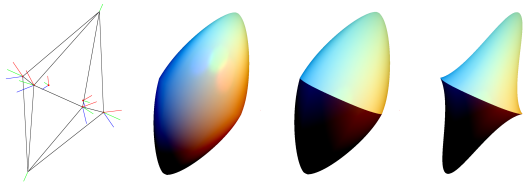
To sum up, an enriched coarse mesh (*ST Mesh* in the remainder) can be defined by using a set of two tables V and T. Each line of table V stores all the data relative to a vertex: the position P, the average normal vector N, the normal discontinuity vector Δ, and the three scalar tags σ, θ, and β. Similarly, each line of table T stores only the three vertex indices (i, j, k) relative to a triangle.

3. Mesh generation

3.1. Combining shading and smoothing

As said above, our technique is strongly based on the PN-triangles presented by Vlachos et al. The reader unfamiliar with this work may refer to [VPBM01] for details on the construction of PN-triangles.

In order to obtain a coherent effect of the shape parameters defined above, their influence has to be accounted both for the shading and the geometry of the surface generated during the rendering process. As shown in Figure 3, in the case of a sharp crease, this approach ensures a coherent behavior, both on the silhouette and at the interior of the object.



(a) (b) (c) (d)

Figure 3: *(a) Coarse mesh with a ring of vertices tagged as sharp ($\sigma = 0.7$). (b) Result obtained with standard PN-Triangles. (c) Result with sharpness only in shading. (d) Result with sharpness both in shading and geometry.*

The shape factors described in the previous section can now be used to efficiently generate a surface with sharp features. Globally, we can make a distinction between:

- the sharpness value σ which mainly acts on the shading,
- the bias β and the tension θ which mainly act on the silhouette of the object, and so on the underlying geometry.

We propose to generate a coherent shading for ST-Meshes with a *procedural normal map* constructed with the modified normals, and a smoothing algorithm for the geometry that can be formulated as a *procedural displacement map*. Both of them are computed with one triangular Bézier patch, similarly to PN-Triangles. The combination of these two procedural maps produces a realtime piecewise smooth visualization that is accurately controlled by the simple per-vertex scalar tags of the ST-Mesh.

3.2. Generation of the normal field

The normal field constructed across a triangle has to be smooth in the interior of the triangle, continuous across an untagged edge (i.e. without normal discontinuity) and consistent to the discontinuity encoded by the σ values of tagged vertices.

To account for tagged vertices, the three original normal vectors are modified in the following way: since Δ_i represents the direction of the discontinuity at vertex i, we define $N_i'^+$ (resp. $N_i'^-$), by $N_i'^+ = (N_i+\sigma_i\Delta_i)/\|(N_i+\sigma_i\Delta_i)\|$ (resp. $N_i'^- = (N_i-\sigma_i\Delta_i)/\|(N_i-\sigma_i\Delta_i)\|$). The choice of $N_i'^+$ or $N_i'^-$ is made according to the classification of the triangle against the triangle-fan split introduced by a sharp crease (see Figure 1). A linear or quadratic (Bézier) interpolation between these normales can produce a visual smoothness over the refined mesh [VPBM01]. In the remainder of the paper, we will note N_i', the normal of the *current* side of a crease for vertex i.

3.3. Generation of the displacement field

As proposed by Vlachos et al., the displacement field will be computed by defining a triangular Bézier patch. But the shape modifications generated by the scalar tags at each vertex have to be accounted for, when generating the displacement field, so the process has to be slightly modified.

A set of 10 Bézier control points have to be computed to define a cubic triangular patch (see Figure 4), to define the displacement field $b(u,v)$:

$$\begin{aligned} b(u,v) \quad = \quad & b_{300}w^3 + b_{030}u^3 + b_{003}v^3 \\ & +3b_{210}w^2u + 3b_{120}wu^2 + 3b_{201}w^2v \\ & +3b_{021}u^2v + 3b_{102}wv^2 + 3b_{012}uv^2 \\ & +6b_{111}wuv \end{aligned} \tag{1}$$

To simplify the upcoming notations, we propose to decompose each control point as:

$$b_i = d_i + e_i \tag{2}$$

where d_i corresponds to the position of the control point when the patch is in a flat configuration (i.e. all control points are lying in the plane) and e_i is the displacement vector when d_i is projected onto the plane defined by the normal and the position of the closest vertex. As in [VPBM01], we classify the control points into 3 main categories:

- **vertex coefficients**: b_{300}, b_{030}, b_{003}
- **tangent coefficients**: b_{210}, b_{120}, b_{021}, b_{012}, b_{102}, b_{201},
- **center coefficients**: b_{111} is procedurally obtained by the formulation proposed by Farin to ensure quadratic precision [Far01].

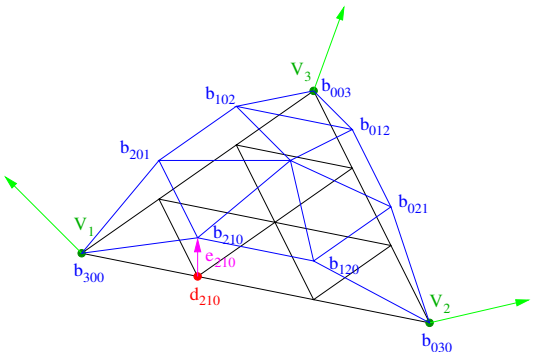


Figure 4: *A cubic triangular Bézier patch. Each control point can be decomposed in a parameter position d_i and a displacement e_i.*

The scalar tags should neither affect the vertex coefficients (as we always want an interpolating surface) nor the center coefficient (as Farin's formulation always maintains a nice shape for the interior of the patch). So, we propose to reduce the geometric expression of the scalar tags only through the tangent coefficients. Moreover, to preserve coherence across triangle boundaries, the scalar tags carried by a vertex will only affect the two nearest tangent coefficients. For example, the scalar tags of V_1 will only affect the coefficients b_{210} and b_{201}.

In the remainder of this section, we consider the case of a coefficient b_i, which is computed using the scalar tags of $V_j = (P_j, N_j, \Delta_j, \sigma_j, \theta_j, \beta_j)$, and the position of the opposite edge vertex P_k. We have also to determinate whether the coefficient is on a sharp edge or not. For this, we use a predicate δ_i which is true if the two relative edge vertices are tagged.

Let $\Pi(p,n,q) = -n.(q-p)$ be the signed distance operator of projection of q onto the plane defined by the point p and the normal n. We can write the Equation 2 as:

$$b_i = d_i + \Pi(P_j, N_j, d_i)N_j$$

with $d_i = P_j + (P_k - P_j)/3$.

For instance, with the coefficient b_{210} of Figure 4, we have $j=1$ and $k=2$; δ_{210} will be true if V_1 and V_2 are tagged, false otherwise. The formulation of this coefficient becomes:

$$b_{210} = d_{210} + \Pi(P_1, N_1, d_{210})N_1$$

with $d_{210} = P_1 + (P_2 - P_1)/3$. Let us now describe how to

modify the geometric definition for a tangent coefficient b_i associated with a tagged vertex V_j.

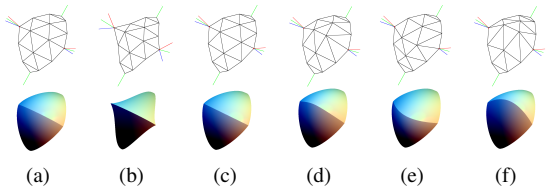


Figure 5: *Transmission of the scalar tags of two vertices to adjacent Bézier patches.* ***(a)*** $\sigma = 0.2,\ \theta = 0,\ \beta = 0$ ***(b)*** $\sigma = 1,\ \theta = 0,\ \beta = 0$ ***(c)*** $\sigma = 0.2,\ \theta = 0.2,\ \beta = 0$ ***(d)*** $\sigma = 0.2,\ \theta = -0.6,\ \beta = 0$ ***(e)*** $\sigma = 0.2,\ \theta = 0,\ \beta = -1$, ***(f)*** $\sigma = 0.2,\ \theta = 0,\ \beta = 1$

Sharpness: To get a consistent silhouette for the refined surface, we have to transmit the sharpness value σ of a vertex to its relative tangent coefficients. This means that b_i has to express the "flatness" of the Bézier patch near the sharp crease (see Figure 5(a) and 5(b)), which is actually the only important aspect for its perception. So, we just have to act on the projection, by constraining b_i to the plane of the "sharp" normal, according to σ, with the following formulation:

$$e_i = (1-\delta_i)\Pi(P_j, X_j, d_i)X_j \text{ with } X_j = \frac{(1-\sigma_j)N_j + \sigma_j N'_j}{\|(1-\sigma_j)N_j + \sigma_j N'_j\|}$$

If $\sigma_j = 0$ we are in the "smooth" case. Otherwise, the modified normal will flatten the patch near the tagged edge by reducing the elevation produced by e_i.

Tension: As usual in tensed Bézier splines, the tension around a vertex V_j will be controlled by the distance between its associated tangent coefficients b_i and its position P_j (see Figure 5(c) and 5(d)). With our formulation, this leads to simply translate d_i before evaluating the projection e_i. We want the tension to be maximal when $d_i = P_j$, so the tangent coefficient will simply be computed by:

$$d_i = P_j + \frac{(1-\theta_j)}{3}(P_k - P_j)$$

Bias: The bias factor is independent of the crease side: two triangles sharing a common vertex V_j of a sharp crease have to conform their Bézier patches in the same direction, defined by the Δ_j (see Figure 5(e) and 5(f)). This time, we want the bias to be expressed only for sharp edges, and we propose to act again on e_i, by using a projection direction that directly takes into account Δ_j. We obtain:

$$e_i = \delta_i \Pi(P_j, N_j, d_i)Y_j \text{ with } Y_j = \frac{N_j + \beta_j \Delta_j}{\|N_j + \beta_j \Delta_j\|}$$

By stitching all together, we obtain the following final formulation for the tangent coefficients:

$$\begin{aligned} b_i &= d_i + e_i \\ d_i &= P_j + \tfrac{(1-\theta_j)}{3}(P_k - P_j) \\ e_i &= (1-\delta_i)\Pi(P_j, X_j, d_i)X_j + \delta_i \Pi(P_j, N_j, d_i)Y_j \end{aligned} \quad (3)$$

The remainder of the process is totally similar to the one used with PN-triangles: the 10 Bézier control points do totally define the continuous displacement field. This field, and the associated normal field, can thus be sampled at a given resolution, to obtain a refined set of triangles that can be sent to the rendering pipeline, or directly used by the GPU with the generic mesh refinement technique of Boubekeur and Schlick [BS05].

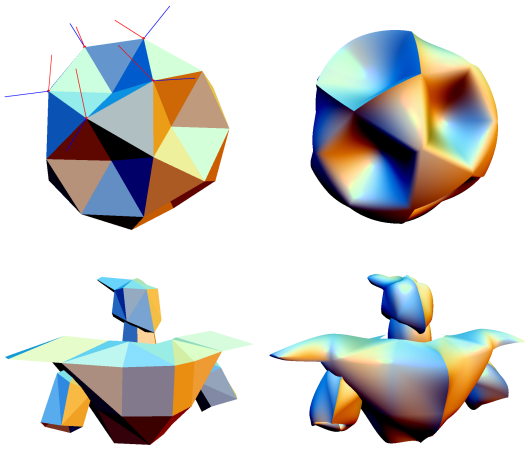


Figure 6: *Original meshes (left) and realtime refinement (right) expressing the scalar tag configuration.*

4. Conclusion and future work

In this paper, we have shown how to easily control some useful local surface properties through a reduced set of scalar shape parameters encoded in a per-vertex basis. This work enriches the original PN-Triangle model, and allows the user to design more complex shapes at a coarse level that will be dynamically refined preserving this shape parameters, which is an interesting property for real-time applications and compression. A first solution for expressing locally sharp edges has been proposed, and our future work will be to investigate the case of multiple sharp edges meeting in the same vertices, and how to locally and efficiently encode this kind of features.

References

[BB83] Brian Barsky and John Beatty. Local control of bias and tension in beta-splines. *Proc. ACM SIGGRAPH*, 1983. 2

[BS05] Tamy Boubekeur and Christophe Schlick. Generic mesh refinement on gpu. In *Proceedings of SIGGRAPH/Eurographics Graphics Hardware*, 2005. 4

[Far01] Gerald Farin. *Curves and Surfaces for CAGD (Fifth Edition)*. Morgan Kaufman Inc., 2001. 2, 3

[VPBM01] Alex Vlachos, Jorg Peters, Chas Boyd, and Jason Mitchell. Curved PN triangles. *Proc. ACM I3D*, 2001. 1, 2, 3

[ZS00] Denis Zorin and Peter Schröder. Subdivision for modeling and animation. *ACM SIGGRAPH Courses Notes*, 2000. 1