Height and Tilt Geometric Texture

Vedrana Andersen[1], Mathieu Desbrun[2], J. Andreas Bærentzen[1], and Henrik Aanæs[1]

[1] Technical University of Denmark
[2] California Institute of Technology

Abstract. We propose a new intrinsic representation of geometric texture over triangle meshes. Our approach extends the conventional height field texture representation by incorporating displacements in the tangential plane in the form of a normal tilt. This texture representation offers a good practical compromise between functionality and simplicity: it can efficiently handle and process geometric texture too complex to be represented as a height field, without having recourse to full blown mesh editing algorithms. The height-and-tilt representation proposed here is fully intrinsic to the mesh, making texture editing and animation (such as bending or waving) intuitively controllable over arbitrary base mesh. We also provide simple methods for texture extraction and transfer using our height-and-field representation.

1 Introduction

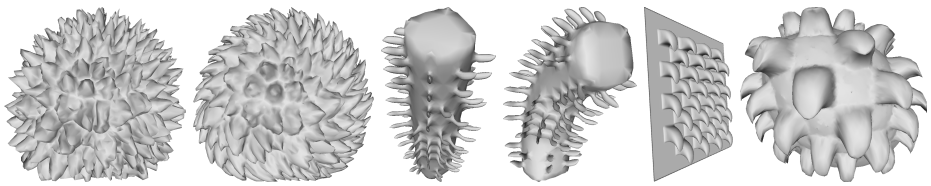

Fig. 1. Examples from our height-and-tilt geometric texture representation. *Left:* A lychee fruit scan is modified to wrap the spikes. *Middle:* Geometric texture applied after a deformation of the base shape. *Right:* A synthetic texture over plane is transferred onto an arbitrary object.

The advent of laser scanners, structured light scanners, and other modalities for capturing digital 3D models from real objects has resulted in the availability of mesh with complex geometric details at a wide range of scales. Handling this geometric complexity has brought numerous challenges. In this paper, we address the problem of representation and editing of the finest level details known as *geometric texture*. It is important to distinguish this use of the word *texture*

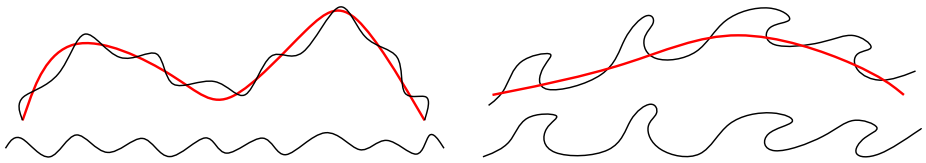

Fig. 2. Limitations of the height field representation of the geometric texture. Of the two textures only the *left* one can be described as the texture superimposed on the a shape

from *texture mapping* where an image is mapped onto a shape via parametrization. In recent years the use of texture mapping has expanded greatly, and one application of texture mapping is to map geometric texture onto a smooth base shape by means of height map images. This approach often performs adequately, but geometric texture such as thorns, scales, bark, and overhangs simply cannot be described by height fields: a single valued height field is insufficient for these common types of geometric texture, see Fig. 2.

Tangential displacements could be included alongside normal (height) displacements. However, there is no simple canonical basis in which to encode tangent vectors. To produce a basis one might use the partial derivative of a map from parameter domain to the surface, or choose one outgoing edge from each vertex. Unfortunately, these obvious methods are not intrinsic to the shape, requiring either an added parametrization, or an ordering of the edges, and further editing of the geometric texture may suffer from artifacts accordingly.

To deal with full 3D texture, researchers have proposed cut-and-paste [1] and example-based [2] methods, as well as approaches that stretch and fit patches of 3D texture to create complex geometric textures [3]. These methods are also capable of handling weaved textures, or textures of high topological genus. They do not, however, offer intrinsic representations of the texture on the surface, but increase the geometric complexity of the object instead, making use of full-blown mesh editing methods [4].

Contributions. We propose an intermediate type of geometric texture representation, compact and practical, offering a compromise richer than displacement field textures but much simpler than full 3D textures. We will assume that small-scale surface details are easily separable from the base surface, but are not necessarily representable as height fields over the base surface. Our representation adds a *tilt* field to the conventional height field texture representation, with this tilt field being stored using one scalar per edge in a coordinate-free (intrinsic) manner. A resulting height-and-tilt texture model can be used for extraction, synthesis and transfer of a large family of geometric textures. Additionally, we demonstrate that dividing a texture into a height field and a tilt field offers new and intuitive mesh editing and animation possibilities without the computational complexity associated with global mesh editing methods, see Fig. 1.

Related Work. Texture is often an important feature of 3D objects, explaining the abundance and variety of methods proposed to synthesize texture on surfaces [5–7]. The main goal of most texture synthesis algorithms is to synthesize a texture (color, transparency, and/or displacement) onto an arbitrary surface resembling a sample texture patch [8, 9]. Common to these methods is the limitation to textures represented by an image or a scalar displacement field.

While height fields defined over surfaces have been used for many years, newer and richer representations have only started to appear recently. In [10] for instance, fur was modeled through the addition of a tangential displacement to rotate a discrete set of hair strands away from the normal direction. A similar idea based on vector-based terrain displacement maps to allow for overhangs was also proposed for gaming [11].

Tangent fields have also recently been used to control texture growth directions [12, 13]. A convenient, intrinsic representation of tangent vector fields was even proposed in [14], along with vector field processing directly through edge value manipulations.

To overcome the limitations of conventional heightfield-based texture representations, we model geometric texture as a locally tilted height field over the base shape. By storing the height field as scalars over mesh vertices (i.e. discrete 0-forms [15]), and storing the tilt field as scalars over mesh edges (i.e. discrete 1-forms), we obtain an intrinsic, coordinate-free representation of fairly complex geometric textures.

2 Background on Tangent Vector Fields as One-Forms

As we make heavy use of representing tangent vector fields as discrete 1-forms, we briefly review the mathematical foundations proposed in [15, 14].

From vector fields to 1-forms. From a vector field defined in the embedding space, one can encode its tangential part $\mathbf{t}$ to a surface mesh by assigning a coefficient c_{ij} to each edge $\mathbf{e}_{ij}$. This coefficient represents the *line integral* of the tangent vector field $\mathbf{t}$ along the edge. The set of all these values on edges offers an intrinsic representation (i.e. needing no coordinate frames) of the tangent vector field.

From 1-forms to vector fields. From the edge values, a tangent vector field can be reconstructed using, for instance, a vertex-based piecewise-linear vector field. The value of the vector field at a vertex is computed from the coefficients of the incident edges: the contribution of one face f_{ijk} (see Fig. 3, *left*) to the field at the vertex v_i is

$$\mathbf{t}_{ijk}(v_i) = \frac{1}{2A_{ijk}}(c_{ij}\mathbf{e}_{ki}^{\perp} - c_{ki}\mathbf{e}_{ij}^{\perp}) \ , \tag{1}$$

where c_{ij} and c_{ki} are coefficients on edges $\mathbf{e}_{ij}$ and $\mathbf{e}_{ki}$ respectively, and $\mathbf{e}_{ij}^{\perp}$ and $\mathbf{e}_{ki}^{\perp}$ are edges $\mathbf{e}_{ij}$ and $\mathbf{e}_{ki}$ (as 3D vectors) rotated for $\pi/2$ in the plane of f_{ijk} (see the discussion about Whitney edge basis functions in [15]). Averaging these

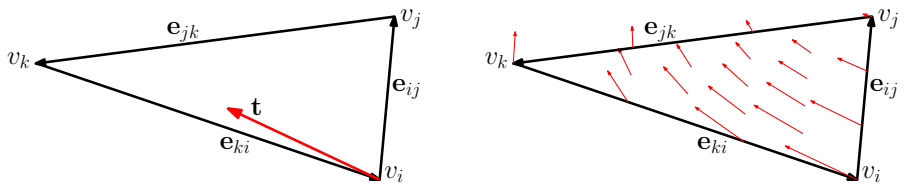

Fig. 3. *Left:* The contribution of the face f_{ijk} to the tangent field at vertex v_i. *Right:* Piecewise linear interpolation of the tangent field

contributions from all incident triangles thus provides a 3D vector per vertex of the mesh.

Least-squares 1-form assignment. The averaging process used in the reconstruction makes the encoding of vector fields by 1-forms lossy: a piecewise-vector field converted into a 1-form may not be exactly recovered once converted back. To provide the best reconstruction of the field from edge coefficients, we do not compute the edge coefficients locally, but proceed instead through a global least squares fit. If we store in $\mathbf{M}$ the reconstruction coefficients defined in (1) (that depend only the connectivity and vertex coordinates of the mesh, not on the tangent vector field), we find the set of edge coefficients $\mathbf{C}$ by solving the linear system

$$\mathbf{M}\ \mathbf{C} = \mathbf{V}\ , \tag{2}$$

where the vector $\mathbf{V}$ contains the coordinates of the input vector field at vertices. Each vertex contributing three equations while there is only one unknown per edge, this system is slightly overdetermined (depending on the genus), and solving it in a least squares fashion yields a very good representation of a tangent vector field over the triangular mesh with little or no loss.

Tangent vector field reconstruction. To transfer a tangential field from one mesh to another we need to evaluate the field on arbitrary point of the mesh surface. For a point P on the face f_{ijk} with barycentric coordinates $(\alpha_i, \alpha_j, \alpha_k)$ associated with vertices i, j and k we get

$$\mathbf{t}(P) = \frac{1}{2A_{ijk}} \left((c_{ki}\alpha_k - c_{ij}\alpha_j)\mathbf{e}_{jk}^{\perp} + (c_{ij}\alpha_i - c_{jk}\alpha_k)\mathbf{e}_{ki}^{\perp} + (c_{jk}\alpha_j - c_{ki}\alpha_i)\mathbf{e}_{ij}^{\perp} \right)\ ,$$

which amounts to evaluating the face contribution to each of the vertices, as in (1), and linearly interpolating those using barycentric coordinates, as illustrated in Fig. 3, *right*.

3 Texture Representation

In the texture representation proposed here, we make the usual assumption that the finely-tesselated textured object comes from a smoother base shape,

onto which a small-scale geometric texture is superimposed without affecting the topology of the base shape. We will first describe how to establish our discrete representation before introducing applications.

3.1 Texture Extraction

Given a finely-tesselated textured object, we must first decide what constitutes geometry (base shape) and what constitutes small-scale texture (displacement from base shape, see Fig. 4, *left*). While this is a notoriously ill-posed problem, many good practical methods have been proposed. In fact, any approach that proceeds through a smoothing of the textured surface while minimizing the tangential drift throughout the process is appropriate in our context. For example, a few steps of mean curvature flow [16] provides a good vertex-to-vertex correspondence between the original textured surface and a smoother version, used as base shape. For more intricate geometries, a multiresolution smoothing strategy such as [17] or a spectral approach such as [18] are preferable (see Fig. 4, *middle*). Alternatively, defining or altering the base shape by hand might be appropriate if specific texture effects are sought after or if the condition mentioned in Fig. 4, *left*, is significantly violated.

3.2 Pseudo-height and Tilt

From displacements to heights and tilts. With a base shape available, the displacement of vertex v_s is simply defined as

$$\mathbf{d} = \mathbf{v}_0 - \mathbf{v}_s \ ,$$

where $\mathbf{v}_s$ is a position of the vertex v_s on the base (smoother) shape, and $\mathbf{v}_0$ is the position of the corresponding vertex on the textured surface (see Fig. 4, *right*). Storing this displacement as a vector would require either using three

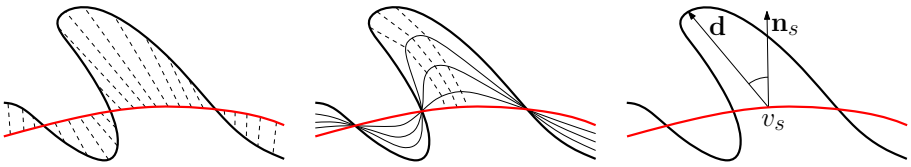

Fig. 4. *Left:* Geometry texture superimposed on objects's base shape in form of vector displacements. The points at the intersections of the textured surface and the base shape have zero displacements. *Middle:* One way of obtaining the base shape in case of non-heightfield texture would be to use the multiresolution hierarchies as in [17] and trace the points through a sufficient number of levels. *Right:* The displacement $\mathbf{d}$ of the vertex v_s can be described in terms of displacement length and the rotation from the normal vector

coordinates, or defining and maintaining an explicit two-dimensional local coordinate frame over the surface. Instead we split the displacement into two fields: a pseudo-height and a tilt, both of which can be represented in a coordinate-free way based on discrete differential forms [15], [14].

The *Pseudo-height* field h represents the signed length of the displacement

$$h = \text{sign}(\mathbf{d} \cdot \mathbf{n}_s)\|\mathbf{d}\| \ ,$$

where $\mathbf{n}_s$ is the normal on the surface at v_s. Our pseudo-height is thus analogous to a typical height field, with values sampled at *vertices* then linearly interpolated across triangles. However we also define a *tilt* field: this is a vector field that defines the tilt (rotation) of the displacement direction with respect to the base normal direction. More precisely, the tilt $\mathbf{t}$ is induced from the displacement $\mathbf{d}$ and the base normal $\mathbf{n}_s$ as

$$\mathbf{t} = \frac{\mathbf{d}}{\|\mathbf{d}\|} \times \mathbf{n}_s \ .$$

Notice that the tilt is a vector in the *tangent space* of the base shape: its direction is the rotation axis for a rotation that transforms the displacement direction into the normal direction and the magnitude of the tilt is the sine of the rotation angle. Therefore, we encode the tilt using the *edge-based* discretization reviewed in Sect. 2. Using the tilt instead of the tangential displacement offers an intuitive description of the texture: the height truly represents the magnitude of the displacement, while the tilt indicates the local rotation of the normal field. We will see that this particular decomposition allows for very simple editing of geometric textures.

In summary, we converted a displacement field into an intrinsic, coordinate-free geometric texture representation

$$\text{texture} = (\text{h}, \mathbf{t}) \ ,$$

consisting of two terms, the pseudo-height h stored as a single scalar per vertex, and the tilt $\mathbf{t}$ stored as a single scalar per edge.

Continuity of height and tilt. Notice that if the condition explained in Fig. 4 is satisfied, our height-and-tilt representation is continuous: the height field vanishes when the textured surface crosses the base shape, while the tilt field approaches the same value on both sides of the surface. However, in practice, one cannot exclude the possibility of having some points that have displacement only in tangential direction, which creates a discontinuity in the height field. To avoid loosing texture information (the "height" of the tangential drift), we use a non-zero sign function in our implementation.

3.3 Texture Reconstruction

Given a base shape and the height-and-tilt texture representation as described above, we can easily reconstruct the textured object. The tilt field $\mathbf{t}$ is calculated

first from the edge coefficients, as explained in in Sect. 2. To obtain the direction of the surface displacement, we then simply need to rotate the base shape normal $\mathbf{n}_s$ around the axis $\mathbf{n}_s \times \mathbf{t}$ by the angle α satisfying

$$\sin\alpha = \|\mathbf{t}\|, \qquad \cos\alpha = \text{sign}(h)\sqrt{1 - \|t\|^2} \ .$$

Our height-and-tilt texture can also be transferred from a source shape to a target shape. We need to define a mapping between the two shapes and sample both the height field and the target shape. Typically, such a mapping between two shapes uses a small number of patches as flat as possible [19], and a mapping between each pair of patches is achieved through, for instance, conformal parametrization of small circular patches. Once such a mapping has been established, our pseudo-height field can be copied from source to target through simple resampling (using, e.g., barycentric coordinates). The tilt can also be transferred efficiently: for each of the target edges, we sample the edge at a number of locations (5 in our implementation), evaluate the tilt vector field (as covered in Sect. 2) at these samples from the map we have between the source and the target, and integrate the dot product of the linearly interpolated vector field over the edge.

Fig. 5 and 6 show three examples of transferring a non-heightfield texture patch to the target mesh by the means of simple resampling.

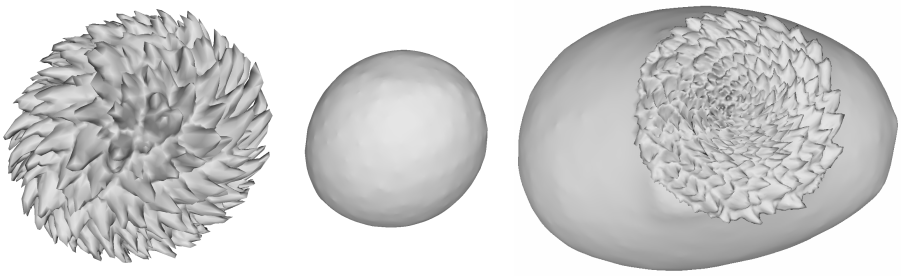

Fig. 5. The texture of the scanned lychee fruit (edited to achieve the whirl effect, *left*) is extracted from the base shape (*middle*) and transferred to the base shape of the avocado fruit (*right*)

4 Applications

We present two types of applications of our height-and-tilt texture representation. For editing and animation, the shape of the object is held constant while the texture on the shape is altered; for deformation and resizing, the shape is deformed and the texture is simply reapplied to it.

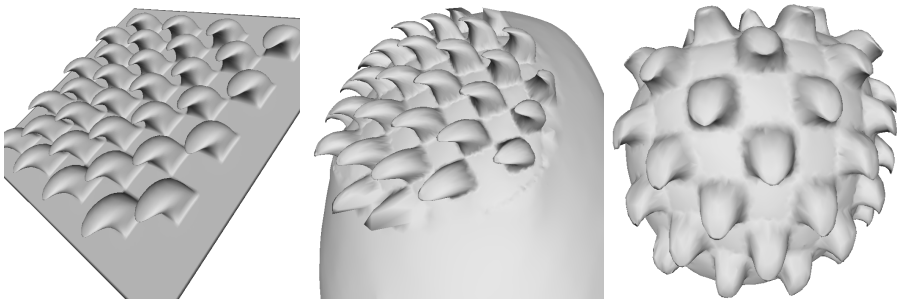

Fig. 6. The synthetical texture (*left*) transferred to the shape of an avocado (*middle*) and to the shape of a lychee fruit (*right*)

4.1 Editing and Animation

Our height-and-tilt texture representation is amenable to a number of simple editing functions. Height and tilt fields can be modified together or separately, which results in new possibilities for geometric texture editing and animation. For instance, we can simulate the effect of spikes swaying on the surface (as if moved by the wind) by changing the texture fields in time. Fig. 7 demonstrates a few examples, such as *set tilt*, which fixes the tilt of the texture; *wrap*, which wraps (bends) the texture spikes; and *wiggle*, which creates a wave-like effect on the spikes. While these operations may not be visually relevant on all textures, they are very effective on spiky textures.

4.2 Deformations and Resizing

Combined with base shape deformation, our representation can also handle a wide range of effects. Fig. 8 exhibits some of the benefits of our approach, where a non-purely height field texture is extracted using a given base shape. The base shape is then deformed, and the texture can be added back in a realistic way. However, since our representation is normal-based, it will still exhibit distortion artifacts for severe bending (i.e. large compared to the scale of the texture). The simplicity of our method cannot (and in fact, is not designed to) handle very complex shape deformation that much more costly Laplacian-based editing methods can [20]. Nevertheless, it alleviates the limitations of height field texture methods while keeping their computational efficiency. In an another example shown in Fig. 9 the base shape has been scaled, but the height-and-tilt texture representation preserves the size and shape of the texture elements.

5 Discussion and Conclusion

We presented a height-and-tilt texture representation to efficiently encode and process small-scale geometric textures over fine meshes. As an extension of

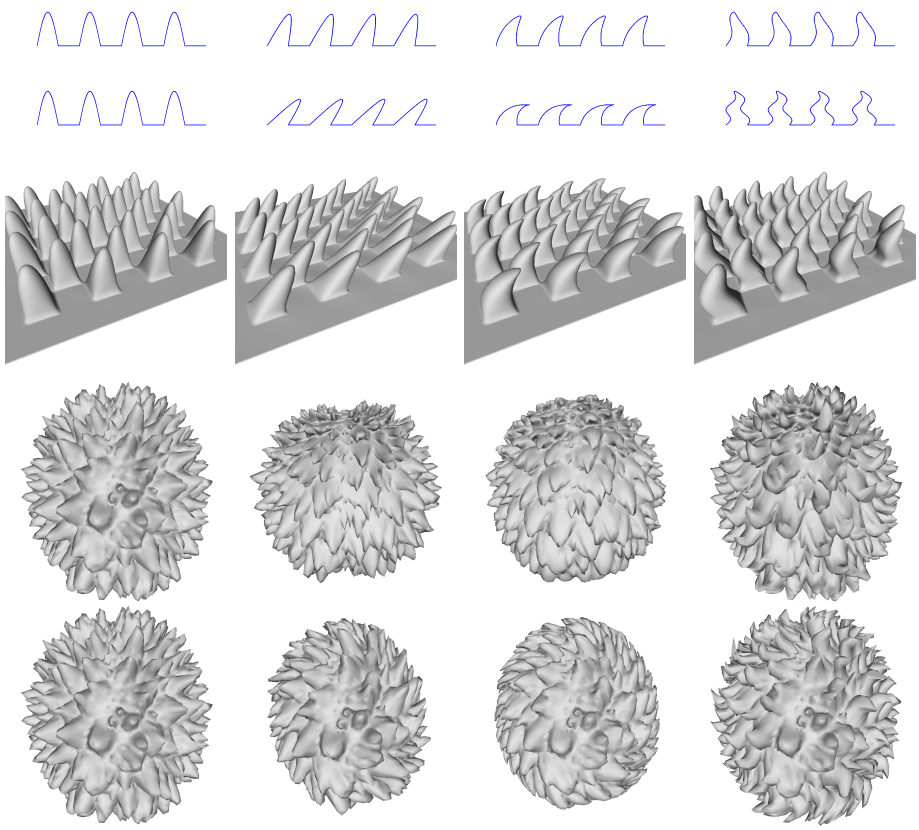

Fig. 7. An example of simple operations on height-and-tilt fields. *Left to right:* A tilt-free texture, set tilt operation, wrap operation and wave operation. *Up to down:* The effect on 2D synthetic texture for two different parameters, on 3D synthetic texture, and on a scan of a lychee fruit for two different directions

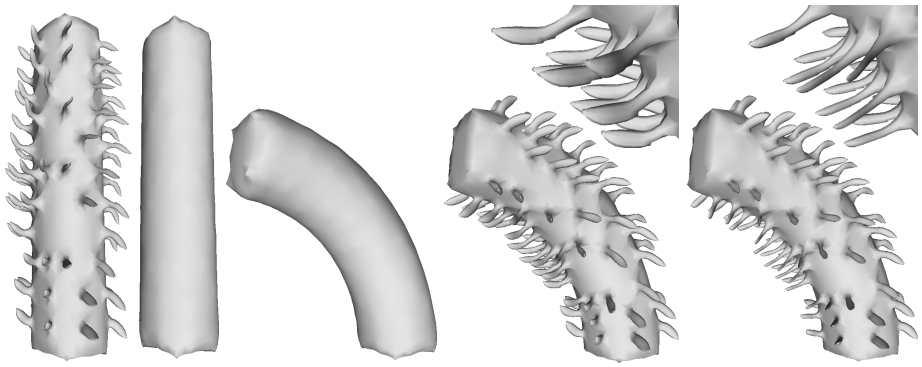

Fig. 8. Our texture representation allows us to extract the texture of the tentacle stick using a (given) base shape. After bending the shape, we can reapply the texture to the shape. On far *right* is the result of applying the space deformation directly to the textured shape. Notice on the enlarged detail that our method does *not* deform texture elements

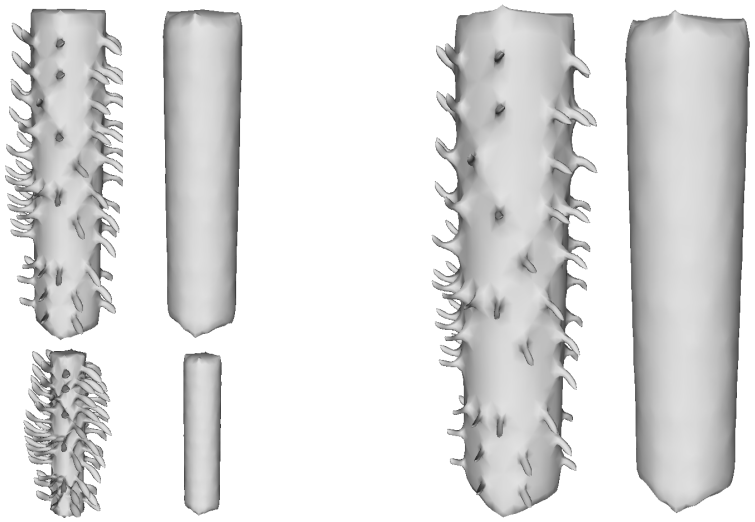

Fig. 9. The original tentacle stick and its (given) shape, *left up*. The shape is then resized (grown by the factor of 1.5 on *right*, shrunk to half size on *left down*) and the texture is put back on it. Due to the texture elements being represented as heights and tilts the size and the shape of the tentacles is not significantly affected by resizing

heightfield-based textures, they share their simplicity (texture editing is achieved only via local computations) and and their intrinsic nature (i.e. they are coordinate-free). Thanks to the added tilt field, a rich spectrum of geometric textures can be stored, edited, animated, as well as transferred between surfaces.

One has to bear in mind some of the present limitations of our method. Firstly, we rely on existing methods to separate texture from geometry. As our notion of texture is richer than the usual height field approach, it is likely that better methods to provide base shapes can be derived. Second, since our representation is normal based, the texture extraction can be sensitive to the smoothness of the base shape. This can be addressed by additional smoothing of the normal field of the base shape prior to texture extraction in our implementation. Additionally, storing the tilt in the tangent field may be, for some applications, inappropriate if the tilt field does not vary smoothly over the surface. To be more robust to non-smoothly varying tilt fields, we utilize the fact that tilt field has maximal magnitude one and constrain the least squares system (2) so that an edge coefficient is not larger than the edge length.

The obvious extension of height-and-tilt texture representation is to synthesize (grow) geometric texture on arbitrary meshes, possibly using the tilt field to control the direction of the growth. Another future endeavor could be to investigate whether we can provide a high fidelity geometric texture with fewer base vertices through field and surface resampling.

Also note the our texture representation is simple enough that a GPU implementation would be fairly easy, allowing for real-time animation of objects displaced with non-heightfield geometric texture or, perhaps more importantly, a system for real time editing of 3D objects with complex geometric texture.

Acknowledgments. This research was partially funded by the NSF grant CCF-0811373.

References

1. Sharf, A., Alexa, M., Cohen-Or, D.: Context-based surface completion. In: SIGGRAPH '04: ACM SIGGRAPH 2004 Papers, New York, NY, USA, ACM (2004) 878–887
2. Bhat, P., Ingram, S., Turk, G.: Geometric texture synthesis by example. In: SGP '04: Proceedings of the 2004 Eurographics/ACM SIGGRAPH symposium on Geometry processing, New York, NY, USA, ACM (2004) 41–44
3. Zhou, K., Huang, X., Wang, X., Tong, Y., Desbrun, M., Guo, B., Shum, H.Y.: Mesh quilting for geometric texture synthesis. In: SIGGRAPH '06: ACM SIGGRAPH 2006 Papers, New York, NY, USA, ACM (2006) 690–697
4. Sorkine, O., Cohen-Or, D., Lipman, Y., Alexa, M., Rössl, C., Seidel, H.P.: Laplacian surface editing. In: SGP '04: Proceedings of the 2004 Eurographics/ACM SIGGRAPH symposium on Geometry processing, New York, NY, USA, ACM (2004) 175–184
5. Wang, L., Wang, X., Tong, X., Lin, S., Hu, S., Guo, B., Shum, H.Y.: View-dependent displacement mapping. In: SIGGRAPH '03: ACM SIGGRAPH 2003 Papers, New York, NY, USA, ACM (2003) 334–339

6. Wei, L.Y., Levoy, M.: Fast texture synthesis using tree-structured vector quantization. In: SIGGRAPH '00: Proceedings of the 27th annual conference on Computer graphics and interactive techniques, New York, NY, USA, ACM Press/Addison-Wesley Publishing Co. (2000) 479–488
7. Hertzmann, A., Jacobs, C.E., Oliver, N., Curless, B., Salesin, D.H.: Image analogies. In: SIGGRAPH '01: Proceedings of the 28th annual conference on Computer graphics and interactive techniques, New York, NY, USA, ACM (2001) 327–340
8. Ying, L., Hertzmann, A., Biermann, H., Zorin, D.: Texture and shape synthesis on surfaces. In: Proceedings of the 12th Eurographics Workshop on Rendering Techniques, London, UK, Springer-Verlag (2001) 301–312
9. Wei, L.Y., Levoy, M.: Texture synthesis over arbitrary manifold surfaces. In: SIGGRAPH '01: Proceedings of the 28th annual conference on Computer graphics and interactive techniques, New York, NY, USA, ACM (2001) 355–360
10. Angelidis, A., McCane, B.: Fur simulation with spring continuum. The Visual Computer: International Journal of Computer Graphics **25** (2009) 255–265
11. McAnlis, C.: Halo wars: The terrain of next-gen. Game Developers Conference (2009)
12. Praun, E., Finkelstein, A., Hoppe, H.: Lapped textures. In: Proceedings of ACM SIGGRAPH 2000. (2000) 465–470
13. Magda, S., Kriegman, D.: Fast texture synthesis on arbitrary meshes. In: EGRW '03: Proceedings of the 14th Eurographics workshop on Rendering, Aire-la-Ville, Switzerland, Switzerland, Eurographics Association (2003) 82–89
14. Fisher, M., Schröder, P., Desbrun, M., Hoppe, H.: Design of tangent vector fields. In: SIGGRAPH '07: ACM SIGGRAPH 2007 papers, New York, NY, USA, ACM (2007) 56
15. Desbrun, M., Kanso, E., Tong, Y.: Discrete differential forms for computational modeling. In: SIGGRAPH '06: ACM SIGGRAPH 2006 Courses, New York, NY, USA, ACM (2006) 39–54
16. Desbrun, M., Meyer, M., Schröder, P., Barr, A.H.: Implicit fairing of irregular meshes using diffusion and curvature flow. In: SIGGRAPH '99: Proceedings of the 26th annual conference on Computer graphics and interactive techniques, New York, NY, USA, ACM Press/Addison-Wesley Publishing Co. (1999) 317–324
17. Kobbelt, L., Vorsatz, J., Seidel, H.P.: Multiresolution hierarchies on unstructured triangle meshes. Computational Geometry: Theory and Applications **14** (1999) 5–24
18. Vallet, B., Lvy, B.: Spectral geometry processing with manifold harmonics. Computer Graphics Forum (Proceedings Eurographics) (2008)
19. Desbrun, M., Meyer, M., Alliez, P.: Intrinsic parameterizations of surface meshes. In: Eurographics conference proceedings. (2002) 209–218
20. Botsch, M., Sumner, R.W., Pauly, M., Gross, M.: Deformation transfer for detail-preserving surface editing. In: Vision, Modeling and Visualization 2006. (2006) 357–364