Paper craft models from meshes

Article *in* The Visual Computer · September 2006

DOI: 10.1007/s00371-006-0067-6 · Source: DBLP

3 authors, including:

George Leifman
Technion - Israel Institute of Technology
13 PUBLICATIONS **655** CITATIONS

SEE PROFILE

The Visual Computer manuscript No.
(will be inserted by the editor)

Idan Shatz · Ayellet Tal · George Leifman

Paper Craft Models from Meshes

the date of receipt and acceptance should be inserted later

Abstract This paper introduces an algorithm for segmenting a mesh into developable approximations. The algorithm can be used in various applications in CAD and computer graphics. This paper focuses on paper crafting and demonstrates that the algorithm generates approximations that are developable, easy to cut, and can be glued together. It is also shown that the error between the given model and the paper model is small.

Keywords Paper-craft models · segmentation

1 Introduction

Many people enjoy crafting models out of paper. The paper layout needed for the task is usually generated by hand or interactively by various companies. The advent of huge libraries of 3D models makes it desirable to perform this task automatically. The challenge is to segment a given model into parts that can be approximated by *developable* surfaces, thus having the property that they can be unfolded onto the plane without stretching, creasing or tearing. Such a segmentation can be utilized in other applications in addition to paper crafting, including texture atlas generation and model fabrication from sheets of material.

A segmentation algorithm that enables the creation of paper-craft models should satisfy the following requirements:

Idan Shatz
Department of Electrical Engineering, Technion
E-mail: shatzi@tx.technion.ac.il

Ayellet Tal
Department of Electrical Engineering, Technion
Tel.: +972-4-8294651
Fax: +972-4-8295757
E-mail: ayellet@ee.technion.ac.il

George Leifman
Department of Electrical Engineering, Technion
E-mail: gleifman@@tx.technion.ac.il

Fig. 1 Paper-craft models

1. Each part must be approximated by a developable surface.
2. The boundaries between these approximations should be easy to cut and glue together.
3. The error defined as a function of the difference between the original model and its paper-craft model, should be small.
4. The number of parts should be small.

Over the last decade, many methods have been proposed for segmenting meshes. They can be divided by the type of segmentation they produce [20]: *texture atlases* and other characteristic patches or *meaningful components*.

The methods for generating texture atlases usually focus on near planar charts [15,11,5,4,6,18,13,19,2,25,24], which are too restrictive for our purpose. Methods that produce other characteristic patches [23] fail in generating smooth boundaries and do not necessarily produce developable segments. Meaningful components, as produced by [21,27,9,26,14,12,17,8], are not

developable, and thus cannot be used for paper-crafting. These general approaches are thus insufficient for our problem.

In the seminal paper of [16], a method is proposed for producing strip-based approximation segmentation of meshes. This method manages to generate very pretty paper-craft models. However, strip-based approximations create long and jagged boundaries that can be difficult to cut and glue, contradicting Requirement 2. In [7], an algorithm for mesh segmentation into nearly developable meshes is introduced and used for creating cloth objects. However, in cloth, differently from paper, the stretch factor tolerates both non-developable segments and non-exact boundaries. Thus, Requirements 1 and 2 are less critical. Moreover, in cloth, many of the details of the objects are lost, and therefore Requirement 3 is often violated.

The current paper presents a novel algorithm for segmenting a given mesh into explicitly developable parts that can be easily cut and glued together, satisfying Requirements 1–4. The algorithm has three key ideas. First, each segment is approximated by a surface that is guaranteed to be developable. Second, the approximations are modified so as to guarantee that neighboring approximations intersect at their boundaries, thus prohibiting stretching. Third, the algorithm extracts the analytical boundaries between the approximations, making the boundaries intuitive and easy to cut and glue. The segmentations produced by the algorithm were used to create several paper-craft models, as illustrated in Figure 1.

The rest of the paper is structured as follows. Section 2 outlines the algorithm. Sections 3–5 elaborate on several stages of the algorithm. Section 6 displays results. Section 7 summarizes the paper.

2 Algorithm Overview

Setup: The aim of the algorithm is to segment a mesh into a small number of segments that can be well approximated by developable surfaces, whose boundaries can be easily cut and glued.

A surface is developable if it has a zero Gaussian curvature at all points. Since this definition does not provide a practical algorithm for generating a segmentation, our algorithm uses two types of surfaces known to be developable: a planar surface and a conic surface, following [7]. Our general scheme, however, can incorporate other predefined types of developable surfaces.

A planar surface is defined by a normal vector n and a constant d, by

$$< n, x >= d. \tag{1}$$

There are several ways to define a conic. We define it as follows: Let c be the center of the cone base, n be the cone axis, d be the distance of a point x from the cone, θ be a constant angle between the cone normals and the cone axis, r_x be the normalized projection of x on the cone base in the axis direction, and n_x be the normal of the cone at x. Then, a conic (n, c, d, θ) is defined by:

$$< n_x, x - c >= d, \tag{2}$$

where r_x and n_x are defined as follows:

$$r_x = \frac{(x-c) - < (x-c), n > n}{||(x-c) - < (x-c), n > n||},$$

$$n_x = r_x \cdot \sin\theta + n \cdot \cos\theta.$$

Note that a planar surface is a special case of a conic, where a conic $(n, 0, d, 0)$ is a plane (n, d). We distinguish between these two types of surfaces because a plane is easier to approximate.

Definition 1 A Segment S_i of a mesh S is a connected sub-mesh $S_i \subseteq S$.

Definition 2 A Segment approximation: Given a segment $S_i \subseteq S$, its approximation E_i is a conic (/plane) associated with S_i.

Given a mesh, the algorithm segments it into disjoint segments $S_1, S_2, \cdots S_K$ whose union gives S, such that each segment S_i is approximated by E_i. The distance between a mesh vertex v and an approximation E_i, is defined as the distance between the vertex and its projection along its n_v direction. (If v is not on E_i, n_v is the direction of the normal of the closest point on E_i.)

$$Distance(v, E_i) = ||v - Proj_{E_i}(v)||. \tag{3}$$

The *total squared error* associated with a mesh and its paper-craft model is defined as the sum of distances from a vertex to the approximations it is associated with, over all vertices:

$$Error(mesh, papermodel) = \sum_{i=1}^{K} \sum_{v \in S_i} (Distance(v, E_i))^2. \tag{4}$$

Algorithm: The algorithm begins with an initial oversegmentation of the mesh into trivial developable segments. This initial segmentation is iteratively modified, by decreasing the number of segments, while increasing the error (Equation 4). Each such iteration approximates the current segments, by fitting each segment to a conic(/plane), using weights specific to our problem.

Once the segmentation is determined, the approximations are modified, in order to accommodate for "good" boundaries. Then, the analytical boundaries between the approximations are computed, therefore not restricting the boundaries to pass through edges of the original mesh. The five stages of the algorithm are briefly described below and explained in detail in the subsequent sections.

1. *Computation of an initial segmentation:* Initially, a planar surface is associated with every face of the mesh, making it a trivial segmentation into developable segments with zero error. Then, neighboring segments are merged if the error (Equation 4) is smaller than a pre-defined error. At the end of this stage, the model is over segmented with a very small error.
2. *Iterative segmentation modification:* The initial segmentation is iteratively modified by applying two operations successively. First, neighboring segments are merged, thus decreasing the total number of segments. Second, the fit between each new segment and its conic approximation is optimized. While the first operation increases the error, the second decreases it. In each iteration, the total error is allowed to increase, until either a pre-defined error or a pre-defined number of segments is reached. This stage is described in detail in Section 3.
3. *Boundary refinement:* The previous stage approximated each segment by a conic (/plane) independently. As a result, neighboring approximations might not intersect each other, or might intersect each other at boundaries that are far from the boundaries between their corresponding segments. The current stage of the algorithm modifies the approximations, so as to take the boundaries into account, as discussed in Section 4.
4. *Extraction of analytical boundaries:* At this stage, the analytical boundaries between the conic approximations are computed. These boundaries are important for two reasons. First, conic edges are easy to cut-and-glue, in contrary to jagged mesh boundaries. Second, analytical boundaries guarantee that neighboring conics meet, which is a vital requirement in paper-crafting (Requirement 2). This stage is discussed in Section 5.
5. *Segment drawing:* After finding the analytical boundaries between segments, they are projected to the plane and printed, adding cuts to conic rings.

Pre-processing: The algorithm described above can be applied to the full mesh. However, two pre-processing steps improve the results. First, a symmetry plane of the model is found, when it exists. The algorithm is applied to half of the model, and the result of the segmentation is duplicated. Second, an existing segmentation into meaningful components (e.g., [9]) is used and the algorithm is applied to each component separately. Both pre-processing steps make paper-crafting more intuitive, since users prefer to work on "semantically meaningful" components, such as the "left and right arm, the "'left and right leg" etc.

The symmetry plane is determined by finding the principal axes of the mesh. There are two types of symmetry: rotation and reflection. Any axis of a rotation symmetry through the origin, as well as the normal of the reflection symmetry plane through the origin, is a principal axis [10]. The algorithm first finds the principal axes of the model using PCA. The Hausdorff distance between the reflected sets of points on both sides of the reflection plane is then used to evaluate the accuracy of reflection symmetry plane. If the distance is smaller than a threshold, the plane is used as a symmetry plane. Otherwise, the algorithm is applied without symmetry.

The following sections elaborate on Steps 2, 3 and 4.

3 Iterative segmentation modification

This stage (Stage 2) applies an iterative region growing approach, using a variant of the K-means algorithm [3]. Every iteration of the algorithm performs the following operations:

1. The faces of the mesh are re-distributed into the current segments. This is done by assigning each face to the conic approximation that best fits it. (In the first iteration, the approximations found in Stage 1 are used.)
   The error associated with a face is defined as a function of its distance from the approximation and the distance of the normals.
2. Each segment is re-approximated by a plane or a conic, using a standard non-linear squared optimization procedure [22].
   Then, the algorithm goes back to Step 1, until either a pre-set number of iterations (10) is reached or until the error does not change from the previous iteration. In this case, the algorithm proceeds to Step 3.
3. Neighboring segments are merged and approximated, until the current error bound is reached.
   The error bound is increased and the algorithm goes back to Step 1.

Below, we elaborate on each of these operations.

Step 1: A face is assigned to a segment if the average distance between its vertices and the segment's approximation is small. The distance depends both on the projection distance (Equation 3) and on the match between the face normals and the approximation, which was empirically found to be important. Specifically, let E_i be the approximation of a given segment S_i and f be the face whose distance to E_i we wish to compute. Then,

$$dist(f, E_i) = NormDiff(f, E_i) \cdot \sum_{v \in f} Distance(v, E_i), \quad (5)$$

$$NormDiff(f, E_i) = 1 + \lambda \sum_{v \in f} (1 - | < N_{E_i}(v), N(f) > |),$$

where $N_{E_i}(v)$ is the normal of E_i at the projection of v on E_i, $N(f)$ is the normal of f, and λ is a user-defined parameter. This step is performed in a Breadth-First Search (BFS) manner, thus guaranteeing connectivity.

Step 2: Given a set of vertices associated with normals, the optimization function attempts to fit both the best conic and the best plane to this set. The surface having the smaller error is chosen. Each face f is assigned a weight $\omega(f)$, which is the normalized area of the face. The area "hints" to the importance of the face vertices. The function optimized in this step is given by:

$$argmin_{n,d,c,\theta} \sum_{f} (\omega(f) Dist(f, E(n,d,c,\theta)))^2 \quad (6)$$

Step 3: To decrease the number of segments, neighboring segments are merged. The segments selected for merging are those whose merge causes the smallest projection error (Equation 4).

4 Boundary refinement

When two segments intersect in a boundary along edges of the mesh, there should be an analytical boundary between their approximations, close to the boundary edges. However, as the error between an approximation and its corresponding segment grows, it might not be satisfied, as illustrated in Figure 2. It is clear that approximating each segment separately cannot suffice.

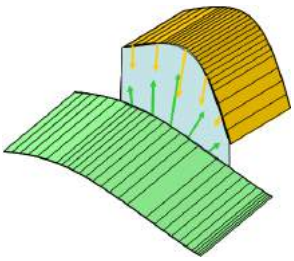


Fig. 2 The problem with boundaries

The challenge is to bring the analytical boundary between adjacent approximations as close as possible to the boundary between their corresponding segments. This is done by considering the boundaries in the optimization process. The distance between the projections of boundary vertices on the adjacent approximations, is added to Equation 6 and each approximation E_i is optimized by:

$$argmin_{n,d,c,\theta} \sum_{f \in E_i} (\omega(f) \cdot Dist(f, E(n,d,c,\theta)))^2$$

$$+\beta \sum_{u \in boundary} (\bar{\omega}(u) ProjError(u))^2. \quad (7)$$

In Equation 7, u is a boundary vertex adjacent to Segments S_i and S_k. Denote the projection of u on E_i (E_k) by u_i (u_k). We define the projection error of u by $ProjError(u) = ||u_i - Proj_{E_i}(u_k)||$. ($\bar{\omega}(v)$ will be defined later.) Note that this second term is 0 when the approximations are perfectly aligned.

At every optimization iteration, the algorithm increases the value of β. (When $\beta \longrightarrow \infty$, the approximations will meet, but the projection error will grow.) The algorithm terminates when β gives a sufficiently small error, so that the boundaries of the approximations are close. In the implementation, β is initialized to 0 and increased by 0.1 at every iteration, where convergence is reached after 10-20 iterations.

Note that this stage of the algorithm is similar to the previous stage (Section 3). Both perform the optimization described in Equation 7, with the only difference being the value of β. However, it is important to perform these two stages separately since the addition of the second term ($\beta > 0$) requires a stable segmentation.

"Flat boundaries": When the boundaries of adjacent approximations have similar normals, their analytical boundary becomes unstable. This situation, which might be counter-intuitive, is illustrated in Figure 3. The approximations are drawn in yellow and green and the analytical boundary between them in red. The blue region demonstrates how the red boundary moves when the yellow approximation changes slightly. It is shown that the blue region increases when the normals at the boundary of the approximations are similar.

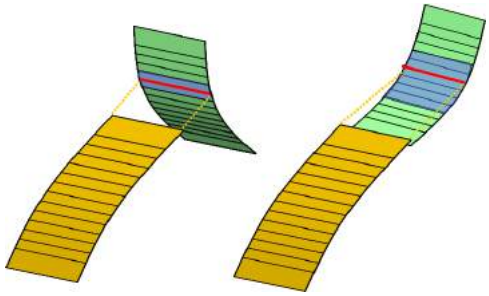


Fig. 3 Sharp (left) vs. flat (right) boundary

To understand this behavior, assume that we are given two planar surfaces p_1 and p_2, having normals n_1 and n_2, respectively. Assume also that p_1 moves in the direction of its normal n_1 by Δd_1, as illustrated in Figure 4. The movement of the boundary between p_1 and p_2, is described by Equation 8

$$\Delta d = \frac{\Delta d_1}{\|n_1 \times n_2\|}. \quad (8)$$

A boundary is called *flat* when $\alpha \approx 0^\circ$ ($\|n_1 \times n_2\| \approx 0$). It can be seen from Equation 8 that when $\alpha \approx 0^\circ$, a small movement of p_1 (Δd_1) will result in a large Δd. Therefore, the projection error of a flat boundary vertex (Δd_1) will cause a large error in the analytical boundary (Δd). To handle it, the weights $\bar{\omega}$ in Equation 7 should

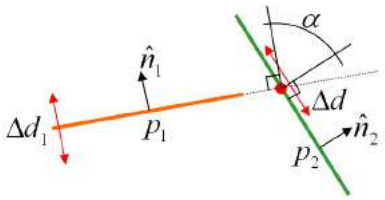


Fig. 4 p_1 moves by Δd_1 in direction n_1

depend on Δd. In our implementation, the weight of a boundary vertex u is set to

$$\bar{\omega}(u) = \frac{1}{\|n_1 \times n_2\| + \epsilon}.$$

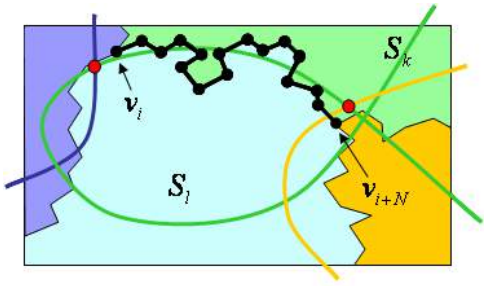
5 Analytical boundary extraction

At the end of Stage 3, the projections of the boundaries of two adjacent segments on their approximations, are close to each other. However, the exact analytical boundary between the approximations, has not yet been extracted. This is the goal of the current stage (Stage 4).

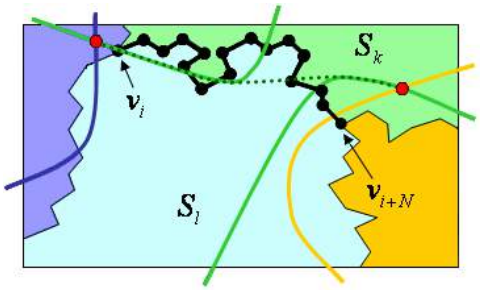
Given two adjacent mesh segments S_l and S_k and their approximations E_l and E_k, respectively, computing the analytical boundary between E_l and E_k can be done analytically, by calculating the intersections between conics. However, three issues need to be addressed. First, an analytical boundary might consist of several disconnected curves, out of which only the relevant curve should be extracted. Second, out of the relevant curve, only a sub-curve should be extracted, as illustrated in Figure 5(a), where the boundary of the approximation of the cyan segment consists of sub-curves of the green, yellow and blue analytical boundaries. Third, in case of "flat boundaries", the analytical boundaries might diverge, as demonstrated by the green curves in Figure 5(b). While the first two issues can be handled analytically, the third cannot.

Therefore, instead of computing the boundaries analytically, the following procedure is utilized.

1. The analytical *branching points* – the intersection points between three approximations – are computed. These are the red points in Figure 5. Every pair of two consecutive branching points are considered the endpoints of a sub-curve that will build up the complete boundary. If two approximations intersect in a full ring (i.e., the boundary between the upper leg and the lower leg of the dino-pet in Figure 7), any point on the ring can be chosen as the first and the second endpoints.
2. We are given two endpoints $\bar{v}_i$ and $\bar{v}_{i+N}$ on the boundary, their corresponding vertices on the mesh v_i and v_{i+N}, and the set of vertices $v_i, v_{i+1}, \cdots, v_{i+N}$ between them. The curve between $\bar{v}_i$ and $\bar{v}_{i+N}$ is computed by iterating on the following three steps until convergence. At time t,



(a) Analytical boundaries in green, blue and yellow



(b) Disconnected analytical boundary

Fig. 5 Analytical boundaries

   (a) $\forall j,\ i \le j \le i+N$, project $v_j(t)$ onto E_l, to get $\hat{v}_j(t)$.
   (b) $\forall j,\ i \le j \le i+N$, project $\hat{v}_j(t)$ onto E_k, to get $\tilde{v}_j(t)$.
   (c) Smooth by setting $v_j(t+1) = \frac{\tilde{v}_{j-1}(t)+\tilde{v}_j(t)+\tilde{v}_{j+1}(t)}{3}$.

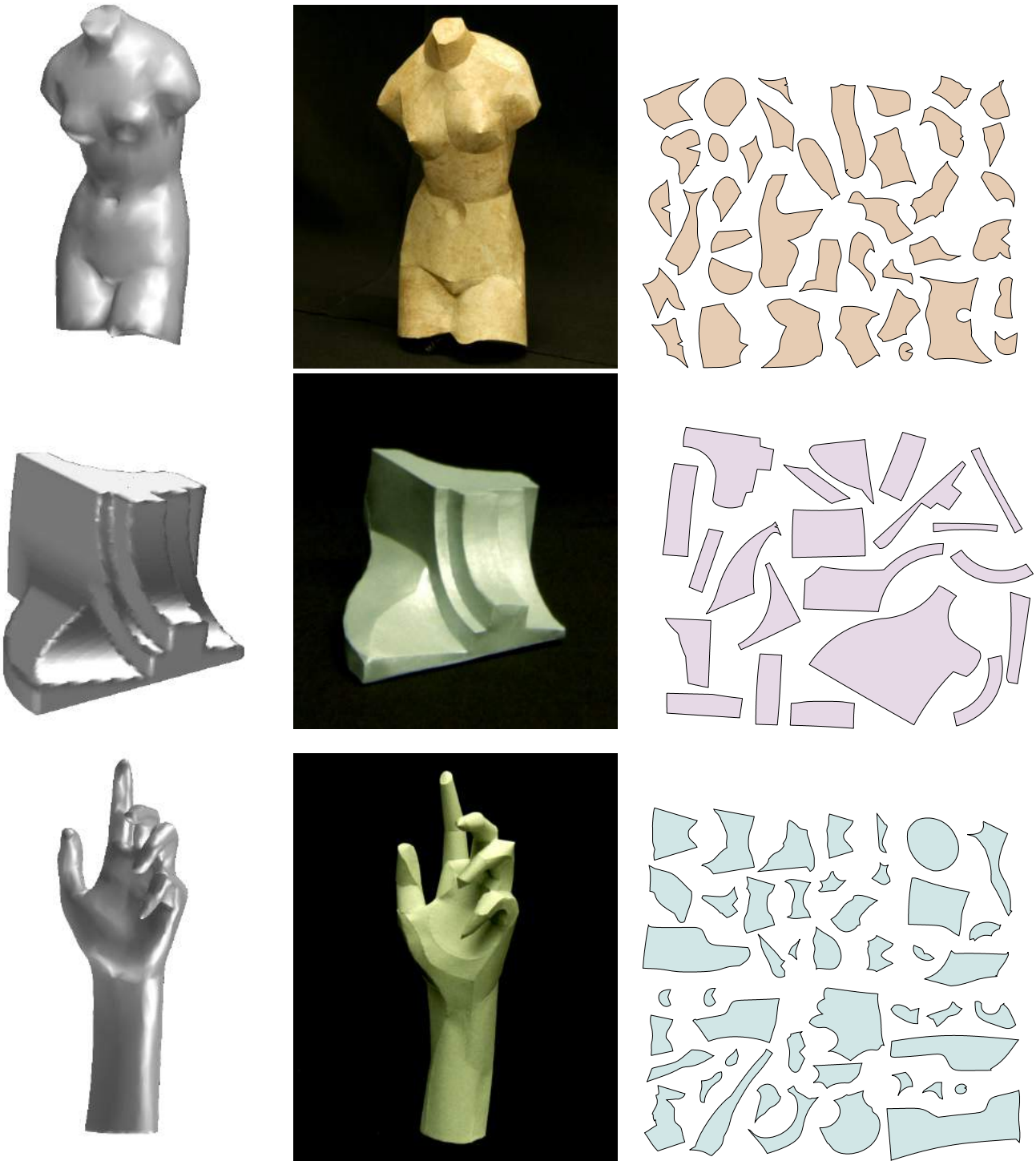
   The first two operations move the current mesh boundary vertices toward the analytical boundary. Convergence is proved in Appendix A. The last operation smooths and shortens the boundary by averaging the points.

Intuitively, the convergence speed depends on the angle between the boundaries. For sharp boundaries, operations (a) and (b) quickly converge to the analytical boundaries, while for flat boundaries, this convergence is slower.

In the special case, where the analytical boundary between two segments consists of several disconnected boundaries (Figure 5(b)), the boundary we are seeking should smoothly connect the two disconnected analytical boundaries (the dashed black line in Figure 5(b)). This case is handled automatically by operation (c). Another desirable outcome of this operation is the collapse of small loops on the boundary.

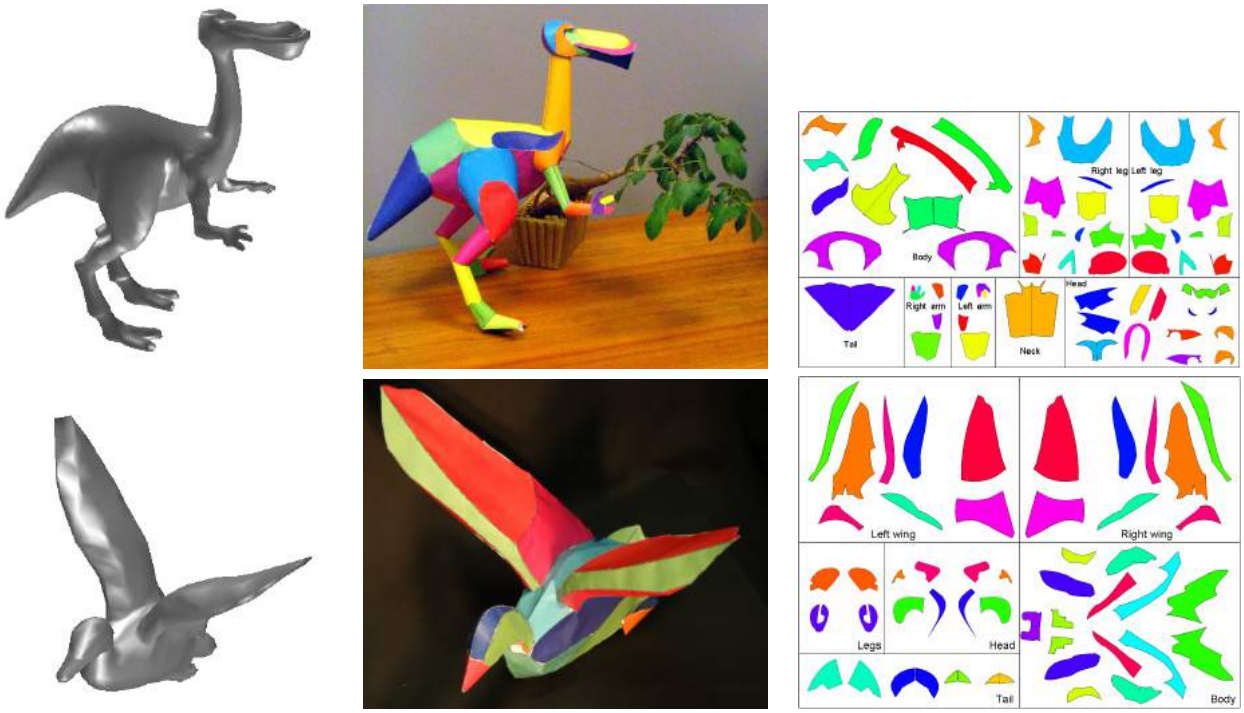
6 Results

Figures 6–7 present some results of the algorithm. It can be seen that the requirements set in Section 1 are satisfied. (1) Each part is developable and can be unfolded



(a) original model (b) paper-craft model (c) part layout

Fig. 6 Paper-craft models of Venus (700 faces), a CAD model (11K faces) , and a hand (25K faces)



(a) original model (b) paper-craft model (c) part layout

Fig. 7 Paper-craft models of the dino-pet (3.5K faces) and the duck (1100 faces)

(a) Our algorithm (b) [Mitani & Suzuki] (c)[Julius et al]

Fig. 8 Comparison

into a paper (Figures 6–7(c)). (2) The boundaries go along piece-wise conic edges and are easy to cut and glue. (3) Visually, the difference between the paper-craft model and the mesh is pretty small. Below we present the measured error. (4) The number of pieces is relatively small.

In Figure 6, no pre-processing was applied to the models, while in Figure 7, both symmetry and segmentation into meaningful components were used in a pre-processing stage. (However, the algorithm identified the cylindrical features of meshes, such as the legs, hands, and neck even without pre-processing.)

In terms of accuracy, our algorithm has the advantage that the error can be bounded, by setting the maximum error in Stages 1 and 2 of the algorithm. In particular, the maximum error was set to 0.05 (for all the models), normalized by the size of the model.

Figure 8 compares the results of our algorithm to those of [16] and [7]. It can be seen that in [7] many of the details disappear, including the front legs, the tail and the protrusion of the rear leg. It can be concluded that the error is large (though it is not supplied). In [16], the paper-craft model presents the details. In this case, the error can be compared numerically. The RMS (root mean square) error measured by Metro [1] between the original model and the paper-craft model (projecting the vertices onto the approximations) of the bunny is 0.0077, compared to 0.0126 reported by [16].

As for the other requirements: Cutting and gluing is easier when cones are used, rather than long triangle strips [16]. It worth noting that other algorithms have attempted to smooth the mesh boundaries [9,7]. However, these boundaries are restricted to the edges of the mesh, and thus are bound to be jagged. In our algorithm, the analytical boundaries solves jagginess.

Finally, our algorithm produces 35 parts for the bunny, while [16] produces 33 parts, thus this aspect is comparable. In [7], only 10 parts are produced, which is acceptable for cloth, but cannot suffice for creating an accurate model from paper.

Our algorithm has several other desirable properties. First, since conics are used, the paper model is relatively smooth. Moreover, the paper model looks smooth even when the input mesh contains very few triangles (e.g., only 912 faces in Venus in Figure 6). Second, when a part is thin (i.e., the hands of the dino-pet in Figure 7), it is modeled by a single planar part. This makes paper-crafting much easier for the user. Third, it can be observed that regions with many details have more segmented than regions with less details. This property helps reducing the number of pieces, while preserving the details of the model. Finally, the use of symmetry, when it exists, makes paper crafting intuitive.

Two parameters need to be set by the user: the maximum error allowed and λ, which specifies the weight given to the normals. The first parameter makes it possible to generate different segmentations of different accuracy for a given mesh, so that the effort is suitable for children at various ages. To produce the paper-craft examples shown in this paper, the error was set to 5% of largest axis of the model's bounding box and λ was set to 5. The exception is the bunny, for which λ was set to 3. Thus, almost no parameter tweaking was necessary.

For the actual gluing of the paper-craft , the gluing instructions are printed on the reverse side of the paper. In particular, the segment number is printed within the segment and the numbers of the neighbors are printed along the corresponding boundaries. Theses numbers guide the user in assembling the parts. Because of the piece-wise smooth boundaries, it is easy to understand where each sub-boundary begins and ends.

7 Conclusion

This paper introduces an algorithm for segmenting a mesh into developable components. The algorithm has three key ideas. First, each segment is approximated by a surface that is guaranteed to be developable. Second, the approximations are modified so as to guarantee that neighboring approximations indeed intersect at their boundaries. This is important when making paper craft models, where stretching is prohibitive. Third, the algorithm extracts the analytical boundaries between the approximations, making the boundaries intuitive and easy to cut and glue.

In addition to satisfying Requirements 1-4, our algorithm has other benefits. First, the user can set parameters that indicate the required level of difficulty. Thus, the algorithm trades-off error for the number of pieces. Second, thin parts are modeled as planar surfaces. Third, symmetry and meaningful components are exploited, which facilitate the understanding of the user. Fourth, the resulting models are piece-wise smooth. Finally, regions with many details are allocated more pieces than regions with fewer details.

The segmentations produced by the algorithm are used to create several paper layouts from existing 3D meshes. The resulting paper-craft models are presented and compared to the results of other algorithms.

In the future, the algorithm can be extended to include other developable surfaces, such as swept surfaces with extrusion. Moreover, although the aim of our algorithm is paper-crafting, it might worth examining whether this method is suitable also for other applications, such as texture mapping, where the developability constraint can be relaxed and therefore the number of charts can be reduced.

The technology described in this paper is patent pending.

References

1. Cignoni, P., Rocchini, C., Scopigno, R.: Metro: measuring error on simplified surfaces. Computer Graphics Forum **17**(2), 167–174 (1998)
2. Cohen-Steiner, D., Alliez, P., Desbrun, M.: Variational shape approximation. ACM Trans. Graph. **23**(3), 905–914 (2004)
3. Duda, R., Hart, P., Stork, D.: Pattern Classification. John Wiley & Sons, New York (2000)
4. Garland, M., Willmott, A., Heckbert, P.: Hierarchical face clustering on polygonal surfaces. In: SI3D '01: Proceedings of the 2001 symposium on Interactive 3D graphics, pp. 49–58 (2001)
5. Guskov, I., Vidimce, K., Sweldens, W., Schroder, P.: Normal meshes. In: SIGGRAPH, pp. 95–102 (2000)
6. Igarashi, T., Cosgrove, D.: Adaptive unwrapping for interactive texture painting. In: SI3D '01: Proceedings of the 2001 symposium on Interactive 3D graphics, pp. 209–216 (2001)
7. Julius, D., Kraevoy, V., Sheffer, A.: D-charts: Quasi-developable mesh segmentation. In: Computer Graphics Forum, vol. 24, pp. 581–590. Eurographics, Blackwell, Dublin, Ireland (2005)
8. Katz, S., Leifman, G., Tal, A.: Mesh segmentation using feature point and core extraction. The Visual Computer **21**(8-10), 865–875 (2005)
9. Katz, S., Tal, A.: Hierarchical mesh decomposition using fuzzy clustering and cuts. ACM Trans. Graph. (SIGGRAPH) **22**(3), 954–961 (2003)
10. Kleppner, D., Kolenkow, R.J.: An Introduction to Mechanics. McGraw-Hill (1973)
11. Krishnamurthy, V., Levoy, M.: Fitting smooth surfaces to dense polygon meshes. In: SIGGRAPH '96: Proceedings of the 23rd annual conference on Computer graphics and interactive techniques, pp. 313–324 (1996)
12. Lee, Y., Lee, S., Shamir, A., Cohen-Or, D., Seidel, H.P.: Intelligent mesh scissoring using 3d snakes. In: Pacific Conference on Computer Graphics and Applications, pp. 279–287 (2004)
13. Levy, B., Petitjean, S., Ray, N., Maillot, J.: Least squares conformal maps for automatic texture atlas generation. In: SIGGRAPH, pp. 362–371 (2002)
14. Liu, R., Zhang, H.: Segmentation of 3d meshes through spectral clustering. In: Pacific Conference on Computer Graphics and Applications, pp. 298–305 (2004)
15. Maillot, J., Yahia, H., Verroust, A.: Interactive texture mapping. In: SIGGRAPH, pp. 27–34 (1993)
16. Mitani, J., Suzuki, H.: Making papercraft toys from meshes using strip-based approximate unfolding. ACM Trans. Graph. **23**(3), 259–263 (2004)
17. Mortara, M., Patanè, G., Spagnuolo, M., Falcidieno, B., Rossignac, J.: Plumber: A multi-scale decomposition of 3d shapes into tubular primitives and bodies. Proc. of Solid Modeling and Applications pp. 139–158 (2004)
18. Sander, P., Snyder, J., Gortler, S., Hoppe, H.: Texture mapping progressive meshes. In: SIGGRAPH, pp. 409–416 (2001)
19. Sander, P., Wood, Z., Gortler, S., Snyder, J., Hoppe, H.: Multi-chart geometry images. In: Eurographics/ACM SIGGRAPH symposium on Geometry processing, pp. 146–155 (2003)
20. Shamir, A.: A formalization of boundary mesh segmentation. In: Proceedings of the second International Symposium on 3DPVT (2004)
21. Shlafman, S., Tal, A., Katz, S.: Metamorphosis of polyhedral surfaces using decomposition. Computer Graphics Forum **21**(3), 219–228 (2002)
22. Taubin, G.: Estimation of planar curves, surfaces, and nonplanar space curves defined by implicit equations with applications to edge and range image segmentation. IEEE Transactions on Pattern Analysis and Machine Intelligence **13**(11), 1115–1138 (1991)
23. Wu, J., Kobbelt, L.: Structure recovery via hybrid variational surface approximation. Computer Graphics Forum **24**(3), 277–284 (2005)
24. Yamauchi, H., Gumhold, S., Zayer, R., Seidel, H.P.: Mesh segmentation driven by gaussian curvature. The Visual Computer **21**(8-10), 659–668 (2005)
25. Zhou, K., Synder, J., Guo, B., Shum, H.Y.: Iso-charts: Stretch-driven mesh parameterization using spectral analysis. In: Eurographics/ACM SIGGRAPH symposium on Geometry processing, pp. 45–54 (2004)
26. Zhou, Y., Huang, Z.: Decomposing polygon meshes by means of critical points. In: MMM, pp. 187–195 (2004)
27. Zuckerberger, E., Tal, A., Shlafman, S.: Polyhedral surface decomposition with applications. Computers and Graphics **26**(5), 733–743 (2002)

A Convergence of analytical boundaries

In Section 5, a three-step procedure was proposed for converging to the analytical boundaries. We show that the first two steps, which project the mesh vertices onto the approximations, indeed converge.

Given a vertex v on segments S_l and S_k, let $v(t)$ be the value of the vertex found in Step 3 at iteration $t-1$ and $Proj_{E_l}(v(t)) = \hat{v}(t)$ be the projection of $v(t)$ on E_l. It is easy to see that the nearest point on E_l to v is the projection of v onto E_l. We define the mutual distance of $v(t)$ from both approximations by

$$Dist(v(t), E_l \cup E_k) = \\ = \max\{Distance(v(t), E_l), Distance(v(t), E_k)\}.$$

Lemma 1 *The mutual distance* $Dist(v(t), E_l \cup E_k) = \\ = Distance(v(t), E_l)$, $t > 1$.

Proof After the first iteration, $v(t)$ resides on E_k, therefore its distance from E_k is zero and the mutual distance must be the distance from E_l.

Lemma 2 *The mutual distance decreases with each iteration, i.e.,* $Dist(v(t+1), E_l \cup E_k) < Dist(v(t), E_l \cup E_k)$.

Proof Let $n_k(v)$ be the normal of E_k at $Proj_{E_k}(v)$, $\alpha_{kl}(v)$ be the angle between $n_k(v)$ and $n_l(v)$, $line_k(v(t))$ be the line from E_k's apex to $v(t)$, $circle_k(v(t))$ be the cross section through $v(t)$ that is parallel to the axis of E_k's, and $v_c(t)$ be the intersection of $line_k(v(t+1))$ and $circle_k(v(t))$. See Figure 9(a).

The distance between $v(t)$ and line $[v(t+1), \hat{v}(t)]$ is $\sin(\alpha_{kl}(\hat{v}(t)))\|v(t) - \hat{v}(t)\|$. Therefore,
$\|v(t+1) - v(t)\| \geq \sin(\alpha_{kl}(\hat{v}(t)))\|v(t) - \hat{v}(t)\|$.

When $\alpha_{kl}(\hat{v}(t)) > 0$, $\|v(t+1) - v(t)\| > 0$. From the triangle inequality:
$\|v(t) - \hat{v}(t)\| + \|\hat{v}(t) - v(t+1)\| \geq \|v(t+1) - v(t)\| > 0$.

As a result, one of the summands must be positive. If the first summand is positive, $\|v_c(t) - \hat{v}(t)\| < \|v(t) - \hat{v}(t)\|$. This stems from the fact that $v_c(t)$ resides on $circle_k(v(t))$ and on $line_k(v(t))$ and thus $v_c(t)$ is the nearest point to $\hat{v}(t)$ on $circle_k(v(t))$. (Figure 9(b) illustrates the case where $\hat{v}(t)$ is external to the cone).

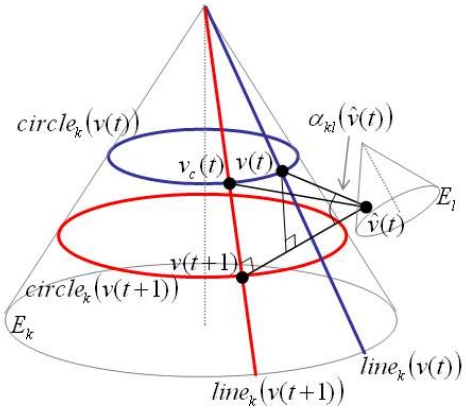
If the second summand is positive, $\|v(t+1) - \hat{v}(t)\| < \|v_c(t) - \hat{v}(t)\|$. This stems from the fact that $v_c(t)$ is on $line_k(v(t+1))$ and $v(t+1)$ is the projection of $\hat{v}(t)$ on E_k, and thus $\{v(t+1), \hat{v}(t), v_c(t)\}$ is a right triangle with a right angle at $v(t+1)$. Consequently, when $\alpha_{kl}(\hat{v}(t)) > 0$,

$$\|v(t+1) - \hat{v}(t)\| < \|v(t) - \hat{v}(t)\|. \quad (9)$$

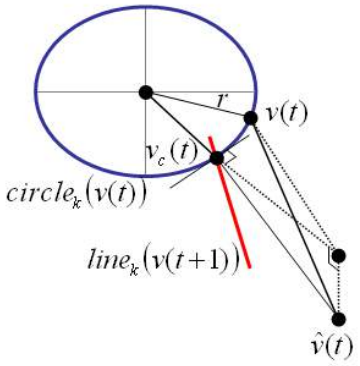
This fact can be used to show convergence:

$$
\begin{aligned}
&Dist(v(t+1), E_l \cup E_k) = \\
(Lemma\ 1) \quad &= Distance(v(t+1), E_l) \\
&= \|v(t+1) - Proj_l(v(t+1))\| \\
(nearest\ point) \quad &\leq \|v(t+1) - Proj_l(v(t))\| \\
(Equation\ 9) \quad &< \|v(t) - Proj_l(v(t))\| \\
&= Distance(v(t), E_l) \\
(Lemma\ 1) \quad &= Dist(v(t), E_l \cup E_k).
\end{aligned}
$$

When $\alpha_{kl}(\hat{v}(t)) = 0$, E_k and E_l are parallel. If they do not intersect, the procedure converges to the nearest points on them.



(a) Definitions



(b) $v_c(t)$ is the nearest point to $\hat{v}(t)$ on $circle_k(v(t))$

Fig. 9 Proof illustration