Developable Surface Processing Methods for 3D Meshes

by

Dan Natan Julius

B.Sc., Tel Aviv University, 2002

A THESIS SUBMITTED IN PARTIAL FULFILMENT OF
THE REQUIREMENTS FOR THE DEGREE OF

Master of Science

in

The Faculty of Graduate Studies

(Computer Science)

The University Of British Columbia

April 10, 2006

Abstract

Developable surfaces are of great significance in computer graphics as they play a key role in many applications involving planar surface parameterization. Industries where 3D objects are constructed from sheets of material such as fabric or metal typically employ these surfaces throughout their design processes. This work introduces a new descriptor for developable mesh surfaces, which provides the means for creating simple and robust tools for detecting, measuring, and approximating developable surface charts in meshes. Based on this descriptor two novel algorithms are proposed. *D-Charts*, an algorithm for mesh segmentation into (nearly) developable charts, and *DCS approximation* an algorithm for approximation of meshes with developable surfaces.

D-Charts uses the proposed descriptor to segment meshing into nearly developable charts for texture atlas generation. By bounding the distortion directly during the segmentation stage, the generated atlases exhibit less distortion for the same number of charts compared to those created using state-of-the-art techniques. In addition to texture atlases, we demonstrate the practicality of this method for industrial applications using the patterns produced by the algorithm to make fabric and paper copies of popular computer graphics models.

The *DCS approximation* method proposed increases surface developability by modifying the geometry, while at the same time keeping the deformed surface as close as possible to the input surface. The method was developed in the context of virtual fabric design, and was combined with a sketching interface to provide a novel design system. The algorithm modifies non-developable meshes generated from sketches into piecewise developable surfaces. These allow straightforward computation of distortion-free texture mapping and automatic generation of 2D patterns for sewing real replicas of the designed garments.

Contents

List of Tables

List of Figures

Preface

Parts of this work have been published, presented or submitted for publication in the following papers and sketches.

- Julius, D., Kraevoy, V. and Sheffer, A. (2005). D-Charts: Quasi-Developable Mesh Segmentation. Computer Graphics Forum, Proceedings of Eurographics 2005, Volume 24, Pages 581-590, Dublin, Ireland, 2005. Eurographics, Blackwell.

- Julius, D., Kraevoy, V. and Sheffer, A. (2005). Real Toys from Virtual Models, SIGGRAPH 2005 Sketch session.

- Decaudin, P., Julius, D., Wither, J., Boissieux, L., Sheffer, A. and Cani, M.P. (2006). Virtual garments: A fully geometric approach for clothing design. Submitted.

Acknowledgements

Co-Authorship statement

Much of this research has been made possible due to the help of others. I am very grateful for their assistance.

The *D-Charts* algorithm described in Chapter 5 was developed together with Vladislav Kraevoy and Dr. Alla Sheffer. Both Mr. Kraevoy and Dr. Sheffer provided substantial input and guidance in means of discussions. Together, they supervised this project, while I performed the actual research and implementation. The *D-Charts* paper published [14] was prepared by Alla Sheffer and myself.

The *DCS approximation* algorithm described in Chapter 6 was developed together with Dr. Alla Sheffer. Dr. Sheffer supervised the project, while I performed the actual research and implementation. The algorithm was then combined with a sketching and fold modeling system developed by Philippe Decaudin, Jamie Wither, Laurence Boissieux, and Prof. Marie-Paule Cani. A paper describing this unified design system was prepared cooperatively and submitted for publication [5].

Chapter 1

Introduction

1.1 Developable surfaces

Developable surfaces are surfaces which can be unfolded (developed) into a plane without introducing any distortion such as stretching or shearing. Simple examples for such surfaces are cylinders and cones (Figure 1.1), while more complex examples of piecewise developable surfaces are shown in Figure 1.2. A sphere is an example of a non-developable surface. These developable surfaces are of great significance in computer graphics, CAD, and manufacturing as illustrated in the following section.

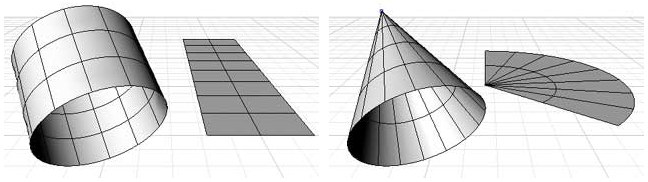


Figure 1.1: Examples of simple developable surfaces and their corresponding patterns.

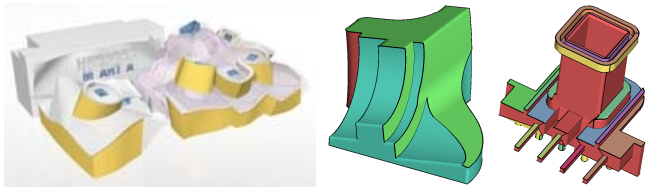


Figure 1.2: Piecewise developable surfaces.

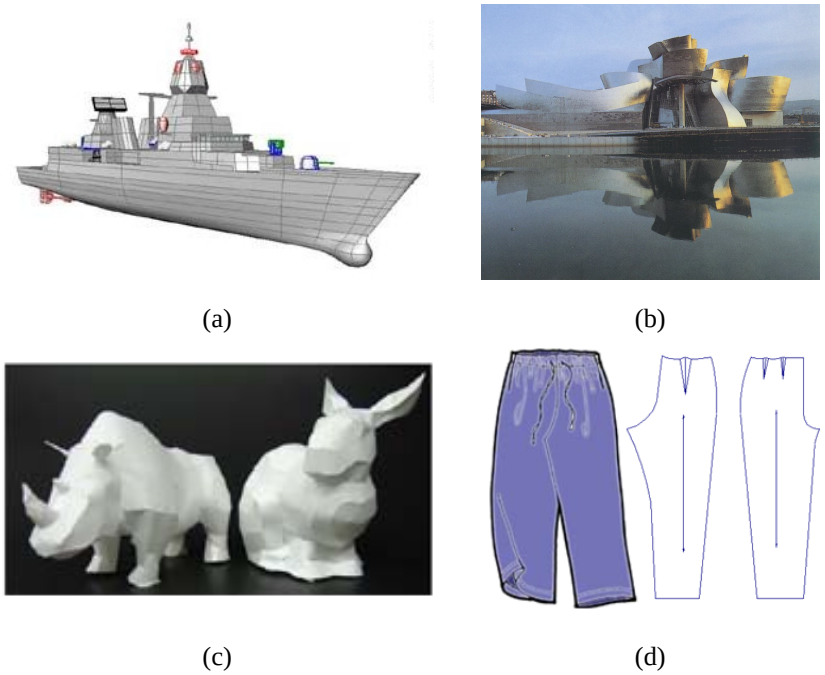
1.2 Applications

In computer graphics developable surfaces play a key role in almost every application involving planar surface parameterization. A planar parameterization of a surface is a bijective mapping from the 3D surface to the 2D plane, which is equivalent to developing the surface into the plane. One of the oldest applications involving surface parameterization is that of texture atlas generation, in which the mesh is first segmented into charts, and then each chart is parameterized onto the plane. When mapping textures, the parameter space is covered with an image, which is then mapped onto the model using the parameterization. With the introduction of programmable GPUs, more general attributes can be mapped onto the model in real time (e.g., BRDFs, bump maps, displacement maps, etc.). It is even possible to represent the geometry of the model in parameter space, leading to the *geometry images* representation [11]. To preserve the texture and other attributes the mapping should be as close as possible to being isometric, and therefore not introduce any distortion. This is possible only if the charts are nearly developable. Boundaries between charts lead to undesirable discontinuities in the mapped texture; hence a good segmentation method must segment the mesh into nearly developable charts while minimizing the number of charts and their boundary lengths.

In manufacturing processes it is common to perform the exact opposite operation of unfolding surfaces, i.e. starting with flat pieces of material, and bending these into the desired shapes. If distortion is avoided during the deformation process, it is clear that the resulting shapes are developable. Since most materials such as metal, wood, and fabric have only limited capabilities of absorbing distortion, developability must be taken into account during the design phase; thus developable surfaces are often used in computed aided design systems (CAD).

Some of the most common examples of manufacturing with developable surfaces are those of ship and aircraft design, architecture, and fashion design (Figure 1.3). When designing ships, the hull is typically segmented into parts which are fitted with large metal sheets. Each metal sheet is shaped using two processes: rolling and heating. The rolling process is used to bend the sheets into cylindrical shapes, and is used in

areas where the desired hull surface is developable. In areas such as the bow, which are not developable, the heating process is combined with the rolling process to shrink the metal sheets along one side, thus creating the required distortion. The heating process parameters are usually determined on the basis of experience and heuristics, thus resulting in a complex and error prone manufacturing process. By improving the design and using mostly developable surfaces, manufactures are able to both simplify and improve the actual construction process [2].



(a) (b)

(c) (d)

Figure 1.3: Applications of developable surfaces.

Similar considerations also apply in building construction when using sheets of material. The most well known architectural designs making explicit usage of developable surfaces are those of Frank Gehry. Gehry's designs are innovative in their usage of developable surfaces as measures of constructibility. Shelden defines constructibility as the ability to form sheet materials without large scale material deformations that would require forming through molding or stamping [40]. While any "free form" surfaces may be constructed from various materials, Shelden notes that these are typically

more costly to create and do not always result in the same quality and desired accuracy. For instance, smooth planar forms are much easier to construct from sheets of metal than from concrete. Moreover, using developability constraints allows Gehry to maintain the correlation between the materials used and their final form [40]. Figure 1.3(b) shows an image of the Guggenheim museum in Bilbao, Spain which is constructed primarily from developable sheets.

In fashion design the usage of developables is even more straightforward. Garments are sewn from patterns, and these are cut from sheets of fabric. Each pattern corresponds to a chart or panel of the garment. When designing the garment, the core requirement is to generate charts that can be cut from flat sheets of fabric and sewn together with minor distortion to form the desired garment. To simplify the sewing both the number of charts and the length and complexity of their boundaries should be minimized. Traditionally these sewing patterns are designed by expert craftsman; however recent advances in the usage of developable surfaces as design constraints will hopefully result in more work being automated.

Similar design considerations also apply to virtual fashion design, i.e. fashion design for virtual characters. This practice is essential in both feature films and real-time applications. In addition to realistic bodies, faces, and hair, designing convincing clothing is a crucial element in achieving character realism. Currently, the modeling of virtual garments is time consuming and requires both technical and tailoring expertise. Even using specialized software, such as MayaCloth [25], users have to design sewing patterns and use physically-based simulation to obtain the garment rest shape. Chapter 6 demonstrates how the developable surface approximation method introduced here simplifies the task of designing patterns for virtual garments.

Paper crafting is yet another example where developable surfaces play a key role. The art of paper crafting is similar to that of Origami, also known as paper folding. The difference between the two, is that Origami designs typically start from a single sheet of paper which is then folded into shape, while paper crafting starts with multiple patterns which are bent into shape and then glued together. Figure 1.3(c) shows some examples of paper crafted models created by Mitani and Suzuki [27]. Since paper does not stretch and it does not shear the simple techniques of bending and gluing used to

craft the models rely on the fact that target model is in fact developable. Worth noting here, is that paper crafting is not used only for artistic or pleasure purposes. Paper models are also commonly used as teaching and visualization aids for math, science and engineering [28].

This thesis focuses on two problems common to these applications and their usage of developable surfaces: Mesh segmentation into (nearly) developable charts, and mesh approximation with developable surfaces. The next sections briefly describe these problems, and their relevance to the mentioned applications.

1.3 Mesh segmentation

Mesh segmentation has numerous applications in computer graphics [37] and manufacturing [2, 40]. As noted previously, this is the first step required when creating texture atlases or patterns from 3D models. Chapter 4 describes a simple descriptor for capturing developable surfaces in meshes. Based on this descriptor, an efficient and robust algorithm named *D-Charts* for segmenting meshes into (nearly) developable charts was created. The algorithm combines geometry processing techniques with tools traditionally used by pattern designers. As a result, compared to recent mesh segmentation methods, for a given number of charts the D-Charts method segments the mesh into more developable charts, leading to less distortion in the generated texture atlases. The algorithm successfully segments mechanical models into their developable components, producing quasi-isometric texture atlases. For other models for which a compact developable segmentation does not exist, the segmentation is based on a user-prescribed developability tolerance.

Based on the segmentation algorithm an automatic method for pattern design, focusing on the design of soft, stuffed toys is introduced. To demonstrate the practicality of the technique, design patterns for several popular computer graphics models were created and then used to construct physical copies of the models out of fabric or paper. The segmentation algorithm and the results of this work have been published in [14].

1.4 Developable approximation

Approximating surfaces with developable surfaces is a useful feature in many design processes. As noted above, developable surfaces play a key role in any manufacturing industry requiring patterns for sewing, metal forming and forging, or any other fabrication applications where 3D objects are constructed from sheets of material. In many industries, it is common to initially focus on the major design requirements, while deliberately ignoring details such as developability which may be corrected later on [45]. For instance, in fashion design the final 2D patterns are developable, however it is hard to imagine that developability plays a key role during the initial sketching of the garment which focuses on the 3D appearance. In such cases, usage of automatic tools can simplify the transition between the design stages. In order to enforce developability while maintaining the original shape as much as possible, the original surfaces must be approximated with developable surfaces. Developable segmentation is not applicable in these cases, since this method relies on creating additional seams across the surface which might be undesired depending on the application. For instance, garments usually have very specific seam lines which are typically symmetric, do not have sharp angles, and are usually positioned on the sides of the body rather than in the front or back. Therefore, an algorithm for approximating a given surface patch with a single developable surface is needed. Chapter 6 describes *DCS approximation*, a method for generating such developable approximations. The DCS approximation method assumes that the original surface as well as the seam lines are provided as input. By taking only the geometric constraints into account the method deforms each surface patch into a ”more developable” surface which is ”close” to the input model. By placing constraints on seam vertices, it is guaranteed that the resulting surface patches remain connected along the seams. Additional constraints may be placed on boundary (or other) vertices to keep them in place at the cost of possibly lower developability.

The method was developed in the context of a virtual garment modeling system. Using a sketching interface proposed by Turquin et al. [43] users generate an initial 3D surface. The users draw the contours and boundaries of the garment directly on a 2D view of the virtual mannequin. This sketch is converted into a 3D surface using the

distance field technique [43]. Given this input, each surface panel enclosed by seams is approximated with a developable surface, while keeping the panels assembled along the seams. This work has been submitted for publication as part of paper focusing on the simplification of virtual garment design [5].

To summarize, this thesis focuses on developable surfaces and their applications in computer graphics. Two related problems of mesh segmentation into developable charts and developable surface approximation are studied, and appropriate algorithms are suggested. Chapters 2 and 3 provide the necessary background and review related work. Chapter 4 explains the two problems of segmentation into developable charts and approximation with developable surfaces targeted in this thesis, followed by a novel method of identifying developable surface patches in 3D meshes. Chapters 5 and 6 introduce the segmentation and approximation methods, followed by the obtained results. Finally Chapters 7 and 8 summarize and discuss conclusions and future work.

Chapter 2

Background

2.1 Definition

Developable surfaces are surfaces which can be unfolded (developed) onto the plane. Mathematically, this means there exists a mapping of the surface onto the Euclidean plane which is locally isometric, i.e. locally length preserving. The mapping does not have to be globally isometric, since developable surfaces may self intersect once unfolded. An introduction to mappings and their properties is found in [7].

The most commonly known property of developable surfaces is that the Gaussian Curvature at all surface points is zero. The following three observations explain why the Gaussian curvature vanishes (A formal proof appears in Theorem 6.1.2 of [33]).

- Isometric surfaces share the same first fundamental form.
- The Gaussian curvature can be expressed in terms of the first fundamental form alone.
- The Gaussian curvature of the plane vanishes at every point.

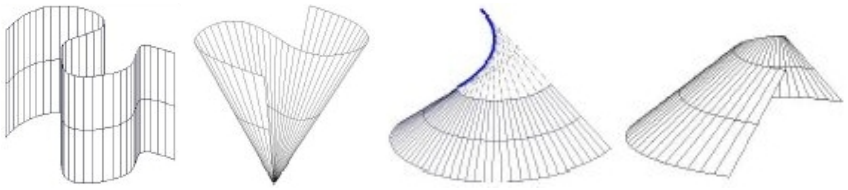
Based on this property it is relatively simple to verify developability of a given surface. For instance, using this property, and the fact that the Gaussian curvature across the sphere is both constant and positive, it is clear that any patch of a sphere, no matter how small, is non-developable. This property, however, does not usually provide enough insight when dealing with construction, manipulation, or unfolding of developable surfaces.

Since it is hard to characterize the entire set of developable surfaces, most research focuses on the subset of sufficiently smooth developable surfaces. The next sections

cover some key concepts regarding these surfaces.

2.2 Developable surfaces as ruled surfaces

Under assumptions of sufficient differentiability, developable surfaces are planes, cylindrical surfaces, conic surfaces, tangent surfaces, or compositions of these surfaces (Theorems 5.1.7 and 6.1.2 of [33]). Peternell [30] illustrates the three later basic surface types as shown in Figure 2.1. These are all *torsal ruled* surfaces, which are a special case of ruled surfaces.



(a) (b) (c)

Figure 2.1: Sufficiently smooth developable surface types [30]: (a) cylinder; (b) cone; (c) tangent surfaces.

A *Ruled surface* is a surface constructed by sweeping a straight line in space. This construction leads to a simple parameterization of such surfaces

$$x(u,v) = b(u) + v\delta(u).$$

where $b(u) \in R^3$ is called the *directrix* or the *base curve* and $\delta(u) \in R^3$ is called the *generator* or *director*. The *rulings* are the straight lines, which are swept in space along the directrix curve, and these are easily apparent by keeping u constant while varying only v.

Following is an outline of Theorem 6.1.3 of [33] which proves that each of the above surfaces are developable. Each of the surfaces may be parameterized as follows:

- generalized cylindrical surfaces – $x(u,v) = b(u) + vp$ where $b(u), p \in R^3$
- generalized conical surfaces – $x(u,v) = p + vb(u)$ where $b(u), p \in R^3$

- tangent surfaces – $x(u,v) = b(u) + vb'(u)$ where $b(u), b'(u) \in R^3$

In order to show that these surfaces are in fact developable, Pottmann and Wallner derive the actual development, i.e. they find an isometric mapping to a plane $s(u,v)$. They then proceed to show that the coefficients of the first fundamental form of $x(u,v)$ in each case are identical to those of $s(u,v)$, thus showing that the mappings are locally isometric.

2.3 Dual representation

An important characteristic of torsal developable surfaces is derived from projective geometry. Since sufficiently smooth developable surfaces are ruled surfaces with vanishing Gaussian curvature, it is easy to show that the tangent planes remain constant for all the points of a generator line. Proof of this is shown by deriving the normal to the surface at every point from the surface parameterizations noted earlier, and verifying that the direction of this normal is constant along the generators. The normal at (u_0, v) is defined by

$$n(u_0,v) = \frac{dx(u,v)}{du}(u_0,v) \times \frac{dx(u,v)}{dv}(u_0,v).$$

By evaluating $n(u_0,v)$ for each of the possible surface types we get the following:

- generalized cylindrical surfaces – $n(u_0,v) = b'(u_0) \times p$;
- generalized conical surfaces – $n(u_0,v) = vb'(u_0) \times b(u_0)$;
- tangent surfaces – $n(u_0,v) = (b'(u_0) + vb''(u_0)) \times b'(u_0) = vb''(u_0) \times b'(u_0)$.

Since the direction of the normal vector does not depend on v, it is clear that the tangent planes are constant for constant values of u, i.e. along generators. Moreover, the tangent planes of the entire surface may be considered as a one-parameter smooth family of planes $T(u)$. This family of planes corresponds to a curve in the dual space as explained next.

Each oriented plane $P \in T(u)$ is uniquely defined by its normal vector of unit length and its signed distance from the origin:

$$P := n_x x + n_y y + n_z z + d = 0.$$

The *Blaschke mapping* ([29]) b maps each oriented plane to a point in the dual space,

$$n_x x + n_y y + n_z z + d = 0 \rightarrow (n_x, n_y, n_z, d)$$

The domain of this mapping is called the *Blaschke cylinder* — $B \subset R^4$. This hypersurface is a quadratic cylinder, and intersections of B with hyperplanes where the fourth coordinate d is constant, are spheres with a radius of one. Mapping each of the planes $P \in T(u)$ results in a smooth curve $t(u) \in B$ corresponding to the smooth family of planes $T(u)$ [29].

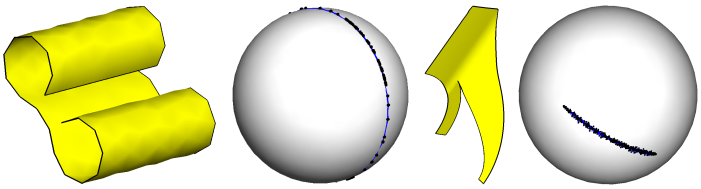


Figure 2.2: Developable surfaces and their corresponding normals maps embedded on the unit sphere.

In Chapter 6 this work uses properties of the normal maps (Gauss maps) of developable surfaces. The normal maps are equivalent to the first three coordinates of points in the dual space. Since the points in the dual space form a curve, it is clear that reducing the dimension by simply removing the fourth coordinate still results in curves on $S3$. Figure 2.2 shows some examples of developable surfaces and the corresponding normal maps plotted on the unit sphere. It is worth noting that this mapping from generator lines to a curve on the unit sphere need not be a one to one mapping. Figure 2.3 shows an example where the curve corresponding to the model is self overlapping. Note however that the curve remains continuous as long as the surface is smooth.

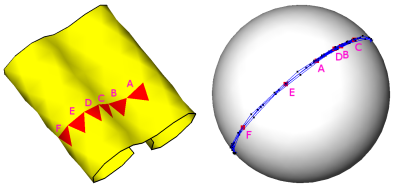


Figure 2.3: A developable surface and the corresponding self overlapping normal map.

2.4 Non-smooth developable surfaces

The previously defined characteristics of developable being ruled surface (Section 2.3) and having a dual representation (Section 2.2) apply only to sufficiently smooth developable surfaces. It is clear however, that the set of smooth developables do not capture the entire set of developable surfaces. A non-smooth developable surface can easily be created, for instance, by taking a piece of paper and crumpling it. The paper remains developable, since it can be flattened out into the plane. As in most previous research, the rest of this thesis focuses mostly on smooth developable surfaces. Nevertheless, the merging method described in Section 5.2.3 does capture straight line creases spanning from one chart boundary to another. The problem of better characterizing and modeling general developable surfaces is left for future research.

2.5 Developability and distortion metrics

As previously noted, developable surfaces can be unfolded into the plane without introducing any distortion. Therefore, mathematically, *developable* is a binary term. Nevertheless, in this thesis the term *developability* is used to quantify how close surfaces are to being developable. A natural measurement is the distortion introduced when parameterizing a surface into the plane. Common metrics to measure the stretch distortion are the $L_2^{Stretch}$ and $L_\infty^{Stretch}$ metrics as defined in [36]. Shearing distortion can be measured using the L_2^{Shear} as defined in [38]. For perfectly developable surfaces where no distortion at all is introduced the stretch metrics are one ($L_\infty^{Stretch} = L_2^{Stretch} = 1$) and the shear metric is zero ($L_2^{Shear} = 0$).

The L_∞ indicates the worst distortion, thus a high value of L_∞ means there is at least one area which is not developable. An exception to this could occur due to the existence of degenerate triangles in the mesh. A single triangle of area ε which is distorted during the parameterization does not indicate a non-developable region in the mesh; however the L_∞ metrics could become very high. The L_2 metrics are averaged distortion metrics. Surfaces which are mostly developable, but include small non-developable areas might still have very low L_2 metrics.

To further complicate the situation, the values of these metrics depend not only on the surface itself, but also on the parameterization method used. While ABF++ [38] results in minimal L_2^{Shear}, stretch minimizing parameterization [35] optimizes the $L^{Stretch}$ metrics. Furthermore, these metrics need not always agree, in the sense that less stretch distortion does not always indicate less shearing distortion and vice versa. Moreover, if one surface has a lower value of $L_2^{Stretch}$ than another, this does not always indicate that the value of $L_\infty^{Stretch}$ for the first surface is also smaller. It is needless to state that it is sometimes hard to compare the developability of two surfaces. As a result, in Chapters 5 and 6 all metrics are presented for each model.

It is worth noting that it is common to use different methods to evaluate the developability of surfaces or quantify algorithm results when no atlas generation is involved. Mitani and Suzuki [27] use Metro [3] to measure the Hausdorff distance between the input surface and the developable estimates they find. Wang and Tang [44] use the maximum Gaussian curvature as a measure of developability. Yamauchi et al. [49] define the area covered by the surface normals on the normal map as their developability metric. Unfortunately, there is no straightforward method of comparing these metrics. For instance a surface could have very high distortion during parameterization, yet be relatively close in the Hausdorff distance-wise sense to its developable approximate. In this thesis the distortion metrics prove to be a more natural choice since the target applications are atlas and pattern generation.

Chapter 3

Related work

3.1 Mesh segmentation

Over the last decade, mesh segmentation has become an important component of many computer graphics algorithms. Shamir [37] provides a review of recent segmentation methods, including a generalized formulation of the segmentation problem. He classifies segmentation methods as being either patch based or, part or feature based, and notes that different methods are more appropriate for different applications. The following two sections review relevant segmentation methods for texture atlas and pattern generation. Other segmentation methods, such as [15, 16], focus on feature-based segmentation. They segment the mesh into charts corresponding to protrusions and other meaningful model components. Typically these charts are very far from being developable and are thus unsuitable for our needs. Methods such as that of Gelfand and Guibas [9] or Wu and Kobbelt [47] are well suited for segmenting CAD point-sets or meshes into kinematic surfaces or primitive structures; however, these surfaces include only a small subset of the developable surfaces.

3.1.1 Texture atlas generation

Over the last decade, many methods have used mesh segmentation to generate texture atlases [13, 18, 22, 24, 34, 49, 50]. Figure 3.1 shows two examples of texture atlases generated by recent state-of-the-art algorithms [34, 50].

Many existing segmentation algorithms focus on the construction of nearly planar charts, since these are clearly developable and will not introduce any parameterization distortion.

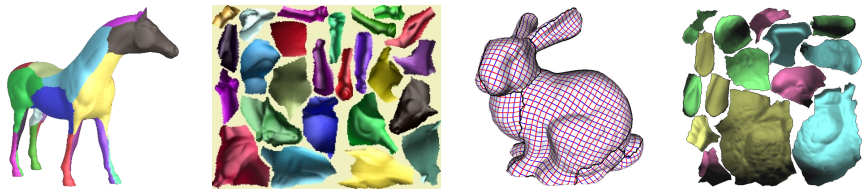


Figure 3.1: Texture atlas examples. (left) Horse model and corresponding charts [34]. (Right) Bunny model with Iso-Lines and corresponding charts [50].

Maillot et al. [24] are one of the first to introduce automatic texture atlases generation methods. In order to minimize the distortion during the parameterization, they suggest segmenting the mesh based on the surface normals. The faces are divided according to their normals into buckets of maximal connected components, such that the normals in each bucket are similar. Adjacent buckets are then merged in a greedy order based on the similarity between their average normal and curvature direction, resulting in a segmentation into nearly planar charts.

Sander et al. [36] segment the mesh based on the planarity and compactness of each chart. They start by assigning each triangle to a separate chart, and then merge adjacent charts in a greedy manner. By enforcing constraints on boundaries when merging they guarantee that patches remain homeomorphic to discs. The process terminates once all merges exceed a user specified tolerance.

Garland et al. [8] use similar means of planarity, compactness and orientation of faces to segment the mesh. The mesh dual graph is constructed and edges are assigned a merging cost. Next adjacent nodes (faces) are merged in a greedy manner to create a hierarchal segmentation of the mesh.

Sander et al. [34] use a Max-Lloyd approach to segment the mesh. Multiple regions are grown around seed triangles based on a greedy approach. Faces are added to charts if their normal is close to the chart normal, and the chart remains compact. After the entire mesh has been segmented, new seeds are calculated and the process is repeated. This method named Multi-Chart Geometry Images (MCGIM) is one of the state-of-the-art methods and is therefore used for comparison with the results of our D-Charts method (Section 5.4).

While planar charts are obviously developable, most developable surfaces are not planar. Thus planar segmentation is too restrictive and results in more charts than are necessary. Following is a review of other segmentation techniques for texture atlases which do not necessarily produce planar charts.

Levy et al. [22] introduce a segmentation method that detects crease lines and generates charts using these lines as boundaries. Charts are then further segmented if the stretch after the parameterization is too high. The method may fail to segment models properly if there are no clear crease lines.

Sorkine et al. [41] simultaneously generate both the parameterization and the cuts. Thus this method can successfully discover developable regions and parameterize them using a single chart. Since this method depends on the order of the triangle insertions it tends to form charts with long and complex boundaries.

Gu et al. [11] construct a single chart out of the input model. They first generate cuts that convert the surface into a disk and then iterate between parameterization and cutting, continuing to add cuts from the current boundary to points of maximal distortion, as long as the distortion is deemed to be too high. Sheffer and Hart [39] also cut the entire surface into a single chart. They detect points of high curvature and connect these by a Steiner tree of cuts, taking visibility into account, so that the cuts go through less visible parts of the model. While both methods result in the minimal number of charts (i.e., one), the chart boundaries tend to be quite long and complex.

The *Iso-charts* method [50] interleaves parameterization and segmentation. Starting with an initial segmentation it repeatedly parameterizes the charts and then segments them if the parametric distortion is high. The authors use a variation of the fuzzy-region cutting approach [16] to generate straight boundaries between the charts. Section 5.4.1 presents a comparison of atlases generated by our method with those generated by Iso-charts.

Yamauchi et al. [49] acknowledge that Gaussian curvature alone is not stable enough in order define a developable segmentation. Instead they use the area of the patch normals on the Gauss map (normal map) as a measure of developability. A flooding approach is used to grow charts from initial triangles marked as seeds, such that the resulting charts have an equal distribution of Gaussian area. A Max-Lloyd it-

eration procedure is used to reposition the seed triangles and regrow the charts in order to further improve the segmentation.

In addition to texture atlas generation, developable mesh segmentation is also relevant to pattern design as demonstrated by the D-Charts algorithm. The next section reviews important contributions to automated pattern design.

3.1.2 Pattern design

Much of the research on pattern generation focuses on unfolding, or parameterization, of given charts. McCartney et al. [26] introduce an automatic unfolding method and demonstrate the impact of darts and gussets on the quality of the unfolding. Darts are stitched tapering folds, generated by cutting a sharp corner out of a pattern. Gussets are triangular inserts added into seams to widen the corresponding regions. Wang [46] proposes another method for the unfolding of free-form surface charts for pattern design by fitting a woven mesh model.

Several methods address segmentation for manufacturing applications. To generate patterns for metal-work, Elber [6] approximates NURBS surfaces by cross-section aligned developable strips. This method requires a very large number of strips to approximate accurately the input models, since only rectangular, ruled surface strips were allowed. Kolmanic and Guid [17] use cross section curves provided as part of the data to segment models into developable strips bounded by these curves in order to generate shoe patterns.

Guthe and Klein [12] introduce a method for atlas generation for NURBS surfaces, which is particularly suitable for pattern design. The charts are formed by stitching NURBS patches together based on distortion considerations. The authors employ a cutting approach similar to that used by [11, 39] using the distortion in the current parameterization as an indicator of where to place cuts. Further cuts are added when the resulting charts overlap in the parameter domain. It is not clear how well these methods will work for irregular meshes.

Mitani et al. [27] introduce a method for pattern generation for making papercraft toys from meshes using strip-based approximate unfolding (Figure 3.2). Since basic

triangle strips can be extremely long, they first segment the mesh as suggested in [22] and then compute strips within each chart. This method results in quite a large number of charts with complex boundaries. As evident by Figure 3.2, using such charts will require complicated methods of cutting and stitching the charts due to their complexity, and is therefore not practical. Therefore, the technique is somewhat specific to paper-crafting; here the core constraint is that paper is incompressible, requiring charts to be truly developable. Since fabric can stretch, the ”stripification” approach is too restrictive for fabric pattern design.

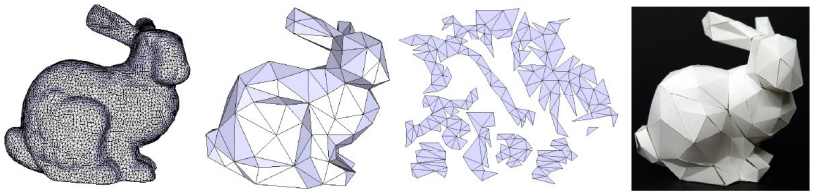


Figure 3.2: Papercraft toys using strip-based approximate unfolding [27].

3.2 Surface approximation with developables

The second problem addressed in this thesis is that of surface approximation with developable surfaces. The relevant research focuses predominantly on the approximation of point clouds in order to identify the underlying analytic developable surfaces [2, 29, 30, 32]. The inputs to these methods are usually sampled from smooth developable surfaces, possibly including some degree of noise. The normal maps of the input surfaces in this case are always very close to being one-dimensional curves (Figure 3.3). It is not clear how these methods could be applied to developable surfaces which correspond to multiple, may be self-intersecting and overlapping, curves in the normal map.

Pottmann and Randrup [32] use cones, cylinders and helical surfaces to approximate given data sets. Chen et al. [2] extend this method to data sets representing more than one single developable element. Using a region growing approach they segment the data into regions which are then approximated separately. Finally, the approxima-

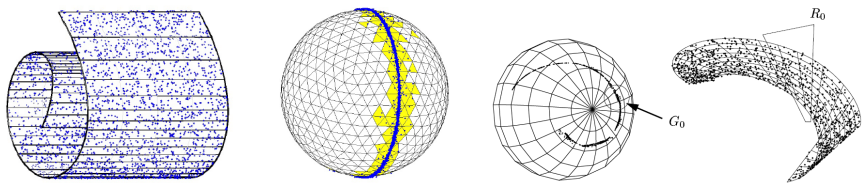


Figure 3.3: Developable surfaces approximating a set of points and the corresponding normal maps [2, 30].

tion elements are smoothly joined to create a G^1 or G_r smooth surface.

Peternell [30] first approximates the tangent planes of the surface in order to create the Blaschke image of the surface. Next, curve approximating techniques ([19]) are used to identify the 1D curves apparent in the Blaschke image. Each curve is then analyzed and the best approximating developable surface among the types defined in Section 2.2 is determined. Peternell acknowledges that the method works best on nearly developable surfaces, such as double curved surfaces with a small secondary curvature or sampled developable surfaces including some degree of noise.

In the past, methods addressing developable surfaces were of great interest for incorporation into NURBS based CAD systems [20, 31, 45]. Early work by Leopoldseder and Pottmann [20] focused on approximating developable surfaces with cone splines in order to incorporate these into existing systems.

Pottmann and Wallner [31] approximate specific types of NURBS surfaces with developable NURBS surfaces. They formulate the problem as finding a set of developable planes to approximate a set of given tangent planes. By optimizing a distance metric defined between two planes, they are able to construct a developable NURBS surface which is close to the input surface.

Wang et al. [45] try to minimize the Gaussian curvature across NURBS surfaces in order to increase developability. By optimizing an energy function accounting for both the amount of deformation and the Gaussian curvature, they improve developability while maintaining shape as much as possible. The authors do however acknowledge that sometimes the improvement in Gaussian curvature is only local.

So far there has been very little research on computing developable approxima-

tions given triangular meshes as an input. Wang and Tang [44] adopt their NURBS surface method [45] to 3D meshes. They translate the developable surface property of zero Gaussian curvature into a requirement that the sum of angles around each vertex be to 2π. They then explicitly minimize the sum of squared differences between the actual sum of angles around each vertex and 2π. The method tends to generate developable meshes with edge lengths which are close to the original. However, since the method is based on a strictly local procedure, it is bound to converge to local minima when operating on input surfaces which are not sufficiently close to being developable. Moreover, since the surface normals are not constrained in any way, they tend to change drastically, resulting in a wrinkled surface — an effect that is usually undesirable.

The following chapters introduce a new method for dealing with developable surfaces and 3D meshed. Two new algorithms which address the segmentation and approximation problems are then presented, and compared to some of the methods noted here in order to evaluate their quality.

Chapter 4

Research contribution

The following chapter describes the two problems addressed in this thesis. Sections 4.1 and 4.2 define the problem of segmenting meshes into nearly developable charts, and approximating meshes with developable surfaces. Next, Section 4.3 explains the key contribution of this work — the *DCS descriptor* and the *fitting error*, which allow for simple detection of developable surfaces of constant slope, and are the basis of the algorithms described in the following chapters.

4.1 Mesh segmentation into developable charts

The first problem addressed here is that of segmenting meshes into developable charts. A formal definition of the generalized mesh segmentation problem is found in [37]. Common to all segmentation problems, the goal is to find a set of disjoint *charts* or *patches*, that cover the entire mesh (Figure 4.1). Specifically in our setting, we would like to minimize the distortion of each chart that occurs during parameterization. As noted in Chapter 1, this may be achieved only if the parameterized surfaces are as close as possible to being developable. Therefore, the first and foremost requirement specific to the segmentation is that each patch be nearly developable.

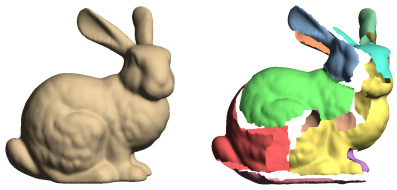


Figure 4.1: Mesh segmentation — disjoint charts cover the entire model.

In contrast to segmentation into planar regions [8, 24, 34, 36], the developability condition alone is typically not sufficient for a meaningful segmentation. Any triangle strip is by definition developable, thus meshes can be ”stripified” using a very small number of triangle strips (Figure 4.2). Obviously this segmentation is not particularly useful for most applications, as it leads to charts with extremely long boundaries. Therefore, in addition to developability of each chart, two additional requirements for a meaningful segmentation are necessary: a small number of charts, and that each chart be compact. Since finding an optimal segmentation satisfying the above requirements is NP-Hard in nature, the D-Charts method proposed in Chapter 5 is an optimization method which searches for local minima.

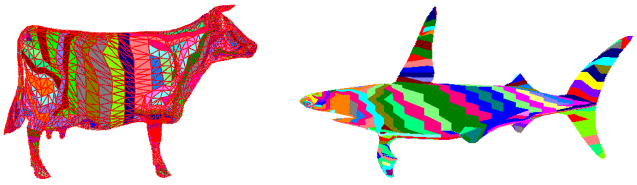


Figure 4.2: Mesh segmentation into triangle strips [48].

This segmentation algorithm uses a region-growing approach. Like [4, 34] it uses an iterative Lloyd scheme [23] to improve an initial segmentation. As in [4], a notation of *proxies* and *seeds* is used to characterize each chart; however in contrast to the previously used planar proxies, a new developable of constant slope (DCS) proxy is used (Section 4.3). A *fitting threshold* is used to enforce the developability of each chart, while additional compactness measures are used to achieve a meaningful segmentation. Another key difference compared to previous Max-Lloyd segmentation methods such as [4, 34], is the fact that the proposed segmentation algorithm does not depend on a the initial number of charts used. The method only requires a rough estimate, while additional charts are added or removed as needed.

To validate the developability of the resulting charts, texture atlases of various models were generated, and the measured distortion metrics were compared to those of previous state-of-the-art methods such as MCGIM [34] and Iso-Charts [50]. In addition, since developable mesh segmentations may be used as the basis for automatic pattern

design, the results were also tested as sewing patterns. Developable segmentations for several popular computer graphics models were created and the resulting patters were then used to sew soft-toys or construct papercraft models of the virtual models.

4.2 Mesh approximation with developable surfaces

The second problem addressed in this thesis, is that of approximating meshes with developable surfaces. In contrast to most previous algorithms, our goal here is not to find the analytical representations of developable surfaces which closely approximate the input surfaces. Instead, our goal is to increase the input surface developability by modifying its geometry, while at the same time keeping the deformed surface as close as possible to the input surface. It is assumed that the input surfaces include all seams, thus the algorithm does not add these. This increases the problem complexity and imposes an additional requirement on the developable surface. Singular surface points, such as cones apices, must not exist, since these require seams connecting them to the boundaries in order to be developed into the plane. Additionally, since most of the input meshes include more than one chart, an additional constraint on boundary vertices is required. Namely, after the geometry is modified adjacent chart boundaries must continue to be perfectly aligned.

To make the charts more developable, we use an approach inspired by the moving-least-squares (MLS) approximation [1, 19, 21]. For each triangle on the surface we find the locally best-fitting developable surface and move the triangle to that surface.

Since previous methods [2, 29, 30, 32] consider only ruled developable surfaces, that is surfaces for which the normal map is a 1D curve, they work well when the input is very close to developable. In our case the input meshes have non-negligible Gaussian curvature, and their normal maps often cover a large area of the normal sphere (Figure 4.3).

To quantify the improvement in developability, each chart is parameterized into a plane and distortion metrics are measured. As described in Section 2.5, the parameterization distortion is a natural metric as the final goal is to generate planar patterns for the surface panels.

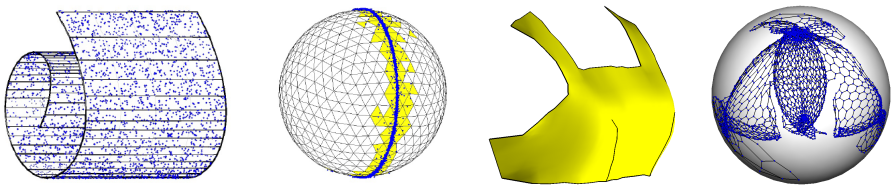


Figure 4.3: (Left) Developable ruled surface approximating a set of points and the corresponding normal maps [30]. (Right) Typical input to our approximation. The normal map shows the triangle normals and the dual graph edges between the triangles.

4.3 Developability and the DCS descriptor

Both the developable segmentation and the developable approximation problems share a common challenge — how to detect developable surface patches in the mesh? The following section describes our novel approach to this task.

While planar charts are developable, they are too restrictive and are therefore not appropriate. Using the vanishing Gaussian curvature as the only means for detecting developable regions in a mesh does not typically provide a viable tool. As noted by Yamauchi et al. [49], it is relatively hard to estimate the Gaussian curvature accurately on meshes and numerical instabilities are often encountered. Therefore, the detection mechanism we propose is narrowed to a subset of developable surfaces — *developable surfaces of constant slope*. The detection procedure is based on a simple observation:

A surface is a *developable of constant slope* if and only if the angle between the normal to the surface at every point and a common axis is constant.

This definition includes cones, generalized cylinders and planes. In the case of a cone or cylinder, the surface normal is at a constant angle or perpendicular to the axis (Figure 4.4). In the case of a plane, the normal is aligned with the axis. This simple condition is used to measure developability.

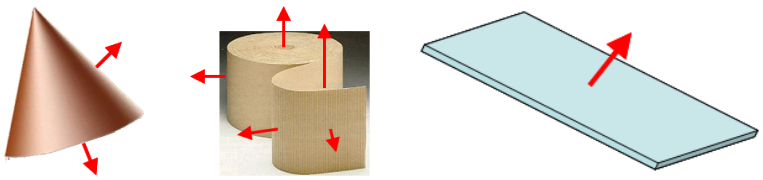


Figure 4.4: Developable surfaces of constant slope

This angle-based condition is the basis for both the segmentation and approximation procedures. In this thesis, we define the *DCS descriptor* for each chart C as the pair $< N_C, \theta_C >$ where N_C is a unit vector representing the axis, and θ_C is the constant angle between the normals to the mesh surface and the axis. To measure how well a given triangle t with a normal n_t fits into a given chart C, we define the *fitting error* as follows:

$$F(C,t) = (\frac{N_C \cdot n_t - cos\theta_C}{2})^2. \tag{4.1}$$

This error measure is significantly simpler than previous metrics used for computing developable regions, such as that of [50] where spectral analysis was needed to achieve a similar goal.

Chapter 5

Mesh segmentation into developable charts[1]

The following chapter describes the *D-Charts* segmentation algorithm [1] [14]. Section 5.1 provides an overview of the algorithm and the *cost function* used to drive the optimization method. Figure 5.3 illustrates the algorithm stages which are explained in detail in Section 5.2. Section 5.3 describes how the resulting charts are then refined into sewing patterns. Finally, Section 5.4 presents the segmentation results as well as patterns used for sewing and constructing soft-toys and papercraft models.

5.1 D-Charts algorithm overview

The D-Charts mesh segmentation algorithm uses a region-growing approach. Like [4, 34] it uses an iterative Lloyd scheme [23], avoiding random initialization issues common to older methods, such as [36, 41]. As in previous methods, a notation of proxies and seeds is used to characterize each chart. The general Lloyd algorithm framework is as follows:

- Grow charts, covering the entire model, based on the current proxies and seeds.
- Compute new proxies and seeds.
- Repeat until the process converges.

[1] A version of this chapter has been published. Julius, D., Kraevoy, V. and Sheffer, A. (2005). D-Charts: Quasi-Developable Mesh Segmentation. Computer Graphics Forum, Proceedings of Eurographics 2005, Volume 24, Pages 581-590, Dublin, Ireland, 2005. Eurographics, Blackwell.

As pointed out in [4], the segmentations computed using the general Lloyd framework depend on the initial positioning of the seeds. To improve the segmentation the authors propose the use of manual operations such as *split* or *merge* to increase or decrease the number of charts, or to ”teleport” them.

In the D-Charts algorithm an automatic procedure based on an initial estimate of the number of charts and a fitting threshold value F_{max} is used to determine the optimal number of charts. The algorithm increases or decreases the number of charts as necessary based on the actual fitting error computed during segmentation. The fitting threshold, as defined in Section 4.3, is also used to ensure that charts remain nearly developable. By fitting each triangle to the chart proxy that represents a developable surfaces of constant slope, the chart developability may be computed and bound.
The steps of our algorithm are as follows:

- **Modified Lloyd Iterations:** Charts are grown using Lloyd iterations while bounding the fitting error by F_{max}. Enforcing this bound ensures that charts remain nearly developable (Section 5.2.1).

- **Hole filling:** As the fitting error is bounded during the growing process, some triangles are not assigned to any chart. During hole filling these triangles are assigned to charts, extra charts are added when necessary (Section 5.2.2).

- **Post-processing:** During post-processing the resulting charts are further improved. First, the boundaries between charts are straightened, without increasing the fitting error. Then, adjacent charts are merged if the combined chart is developable. Finally, seams are cut toward regions of high error, forming darts and gussets (Section 5.2.3).

The algorithm stages are explained in detail in Section 5.2.

5.1.1 Cost function

The main motivation of the segmentation algorithm is to segment the mesh into (nearly) developable charts. However, as noted in Section 4.1 the developability condition alone

is typically not sufficient for a meaningful segmentation. Hence the challenge faced is to segment the mesh into reasonably compact, developable charts.

Developability is measured using the fitting error $F(C,t)$ (Eq. 4.1). To test for compactness when adding a triangle to a chart, two additional metrics are introduced, the first aiming at the generation of relatively "round" charts, the second at the generation of charts with straight boundaries. Both metrics measure the suitability of adding a triangle to a chart when the triangle shares one or more edges with this chart. To measure compactness the following cost is defined

$$C(C,t) = \pi \frac{D(S_C,t)^2}{A_C}, \tag{5.1}$$

where S_C is the seed triangle of the given chart, $D(S_C,t)$ is the length of the shortest path (inside the chart) between the two triangles, and A_C is the area of chart C. For triangles on the boundary of a circle (which is ideally compact) this metric evaluates to one; otherwise, distant triangles are penalized, while close triangles are promoted (Figure 5.1).

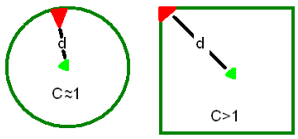


Figure 5.1: Compactness metric.

To promote straight boundaries the ratio between the length of the triangle's edges that are shared with the chart and those that are not is used. The cost function is defined as

$$P(C,t) = \frac{l_{outer}(C,t)}{l_{inner}(C,t)}. \tag{5.2}$$

To combine the developability and compactness metrics, a single cost function is defined. This function measures the cost of adding a triangle to a chart, where the triangle shares one or more boundary edges with triangles inside the chart. The complete cost metric for the triangle is thus:

$$Cost(C,t) = F(C,t)^{\alpha} C(C,t)^{\beta} P(C,t)^{\gamma}. \tag{5.3}$$

The weights α, β, and γ control the importance given to each metric. For all the examples the following values were used $\alpha = 1$, $\beta = 0.7$, and $\gamma = 0.5$. These numbers were found empirically to provide an appropriate balance between developability and compactness. Using lower values of β and γ sometimes resulted in elongated strips or jagged boundaries. Figure 5.2 demonstrates the effect of β when segmenting the sphere.

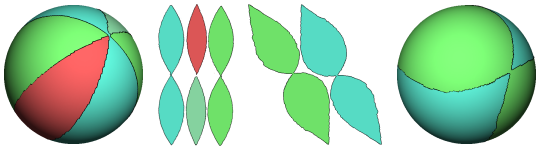


Figure 5.2: Segmentation of the sphere. Left — using $\alpha = 1, \beta = 0, \gamma = 0$. Right — using $\alpha = 1, \beta = 0.7, \gamma = 0$.

5.2 Algorithm stages

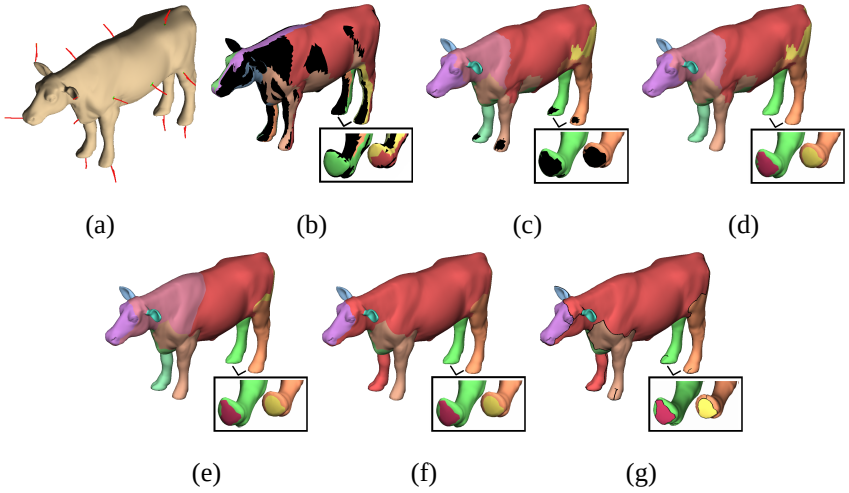


Figure 5.3: Stages of D-Charts on a bull model with a bottom view of the back feet: (a) Seeds (b) Charts after one growth iteration. (c) Final charts. (d) Hole filling. (e) Boundary straightening. (f) Merged charts. (g) Adding seams.

Given the above definitions of a proxy, a fitting error, and a cost function, this section describes the stages of the algorithm in more detail (Figure 5.3).

5.2.1 Modified Lloyd Iterations

Initialization: Given the initial number of charts, the algorithm performs a two step initialization of seeds and proxies. First a seed for each chart is selected. Although in theory any random set of triangles should be adequate (since the following iterations will eventually move the charts into place), in our experience, when extremities exist in the mesh a more refined choice of seeds usually leads to faster convergence with better results. Thus, the distance between seeds is maximized, using the standard farthest point algorithm.

Next, a proxy is calculated for each of the seeds. For each seed three potential charts are examined — the sets of triangles around each of the three seed-triangle vertices. The algorithm calculates a proxy for each chart as explained below (Equation

5.4) and selects the proxy that minimizes the fitting error (Equation 4.1) for the seed triangle. Figure 5.3(a) shows the bull with the initial seed triangles marked.

Growing Charts: Initially each chart contains only a single triangle — the seed. For each seed, each of its adjacent triangles are inserted into a priority queue. For each triangle both its adjacent chart and the cost of assigning it to that chart (Equation 5.3) are specified. As long as the queue is not empty, the triangle with minimal cost is extracted from it and assigned to its specified chart if the following conditions hold:

1. The triangle has not yet been assigned to any other chart (this can occur, since each triangle may have more than one entry in the queue).

2. The fitting error for the triangle and specified chart is below the threshold $F(C,t) < F_{max}$. The fitting threshold ensures that charts remain nearly developable.

Otherwise, the triangle is skipped and the next triangle in order is processed. After assigning the triangle to the chart, its adjacent triangles, which have not yet been assigned to any chart, are inserted into the queue (specifying the current chart and the cost with respect to it). After the process terminates, proxies are recomputed and the growth procedure is repeated. Figure 5.3(b) shows the bull charts after a single iteration.

Note that both $P(C,t)$ and $C(C,t)$ (Eq. 5.1 and 5.2) change as chart C is grown, thus changing the $Cost(C,t)$ (Eq. 5.3) for each triangle. To avoid re-sorting the entire priority queue after every step, multiple queues are used — one per chart. At each step the triangle with minimal cost is found by comparing the first triangle from each queue. This method requires only local updates to each queue after each step. Global sorting is not needed as the order of triangles in each queue in terms of cost is not altered.

Finding new proxies and seeds: The optimal proxy $< N_C, \theta_C >$ for a given chart contains the normalized axis vector and the angle that minimize the weighted fitting error between the conic surface and the chart triangles. The new proxy is computed by solving the following constrained optimization problem:

$$\min_{N_C,\theta_C} \frac{1}{A_C} \sum_{t \in C} A_t F(C,t) \quad \text{s.t.} \quad \|N_C\|^2 = 1, \tag{5.4}$$

where A_t is the area of the triangle t. Knitro [51] is used to obtain the solution. Since only four variables are involved, the solution takes only milliseconds.

After calculating the new proxies, the algorithm selects a new seed S_C for each chart. The seed must fit the proxy well (i.e., with only a small fitting error), and to the extent that is possible, it should be near the center of the chart. To find such seeds the algorithm examines the first k triangles in the chart with minimal fitting error ($k = 10$ in all our examples), and then selects the one closest to the center of the chart. This approach improves the method suggested in [4] since nearly developable surfaces will have a large number of triangles with very low fitting error, resulting in an almost random selection of new seeds.

Termination: The process is terminated when only a small percentage of triangles (below 5%) have been reassigned from one chart to another between two consecutive iterations. While the convergence of Lloyd iterations is guaranteed in the continuous case, this guarantee does not extend to 3D meshes [4]. In practice the algorithm converges in a small number of iterations (fewer than 100) given a reasonable initial estimate of the number of charts alongside a small F_{max}. To guarantee termination users may specify a bound on the number of iterations. Figure 5.3(c) shows the bull after the iterations have converged.

5.2.2 Filling holes

After the Lloyd iterations converge, most of the triangles are classified as belonging to one chart or another. However, due to the use of the error bound F_{max}, some triangles may not belong to any chart (Figure 5.3(c)). All such triangles are collected into connected components which are categorized as either large or small holes, depending on the component area. Small holes are then filled by removing the bound on the fitting error and growing the surrounding charts into the holes (the introduced error is later reduced by adding partial cuts (Section 5.2.3)). Large holes are dealt with by adding additional charts inside the holes. First, all the triangles in the hole are assigned to a new chart; next, an optimal proxy and seed are computed, and finally the chart is ”re-grown” inside the hole as described above. This may result in additional holes; and these are dealt with recursively. This approach leads to better results than spawning new proxies during the iteration phase. Regions of high Gaussian curvature, such as

the tip of the nose on the Horse model (Figure 5.7), repel the growing charts. Even newly spawned charts are pushed away after a few iterations, resulting in additional ones being spawned and an excessive number of charts in total. Figure 5.3(d) shows that the small holes on the top side of each of the bull's feet are filled by adjacent charts, while the larger holes on the bottom are filled by creating additional charts.

5.2.3 Post-processing

The post-processing performs three major operations: it straightens chart boundaries, merges adjacent charts when developability conditions allow, and finally introduces partial cuts into the mesh.

Straightening boundaries: After convergence, boundaries between charts often tend to be jagged, especially in areas where fitting errors between the triangles and each of the two adjacent charts are similar. To improve the segmentation, fuzzy areas between each pair of adjacent charts are defined. Fuzzy areas are the regions along the boundaries in which the triangle fitting error is low with respect to both charts. To find these regions, both charts are grown virtually into one another. Since the fitting error is bounded by F_{max}, only triangles with low fitting errors are reached and marked as belonging to the fuzzy area. Next, each boundary segment between the charts is adjusted. For each segment, first its two end-points are found; then the shortest path, within the fuzzy area, between these points is found, and defined as the new boundary segment. Figure 5.3(e) shows the straightened boundaries on the bull.

While this notion of fuzzy regions is somewhat similar to that used in [16], the boundary straightening algorithm itself is much simpler. The min-cut approach [16] is used only when the boundary is circular, namely when the two charts share an entire boundary loop.

Merging charts: The developability metric (Equation 4.1), captures developable surfaces of constant slope. To capture more general developable surfaces, the algorithm merges adjacent charts if together they still represent a nearly developable surface (Figure 5.4). Using the same framework, for each pair of adjacent charts the algorithm tests to see if the boundary region between them (formed by triangles near the common

boundary) can be approximated by a cylindrical surface. This is a sufficient condition for the combined chart to be developable, and necessary if each chart lies on a different DCS surface.

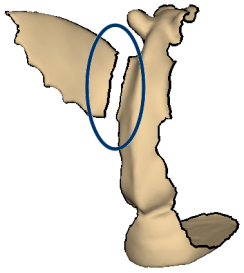


Figure 5.4: Merging adjacent charts without reducing developability.

Given the boundary region B, the algorithm computes a proxy $< N_B, \theta_B = \pi/2 >$ (Equation 5.4). The charts are merged if the average fitting error is below a given threshold η:

$$\frac{1}{A_B} \sum_{t \in B} A_t F(B,t) < \eta.$$

In the mechanical examples (Figure 5.8) $\eta = 1e-5$ was used to keep the charts strictly developable; in all the other examples values of $\eta = 1e-2$ were used. Figure 5.3(f) shows the merged charts.

Darts and gussets: Due to hole filling, charts can contain small regions of triangles with large fitting errors. These regions can cause high distortion or "strain" in the surrounding region during parameterization. To reduce the strain, the pattern design technique of darts and gussets has been incorporated into the algorithm, introducing partial cuts into the charts. These partial cuts are inserted from the boundaries toward the regions with high fitting error. Each seam relaxes the strain in a circular area centered at the tip of the cut, with a radius equal to the length of the seam; thus, elongated high-error regions may require a number of seams. Figure 5.5 shows examples of a dart and gusset in real fabric followed by a dart in virtual pattern created by the algorithm.

During parameterization, cuts corresponding to darts will naturally open up in 2D, while cuts corresponding to gussets will result in overlaps in the 2D parameterization. Therefore, after the parameterization, the overlaps need to be located, the overlapping

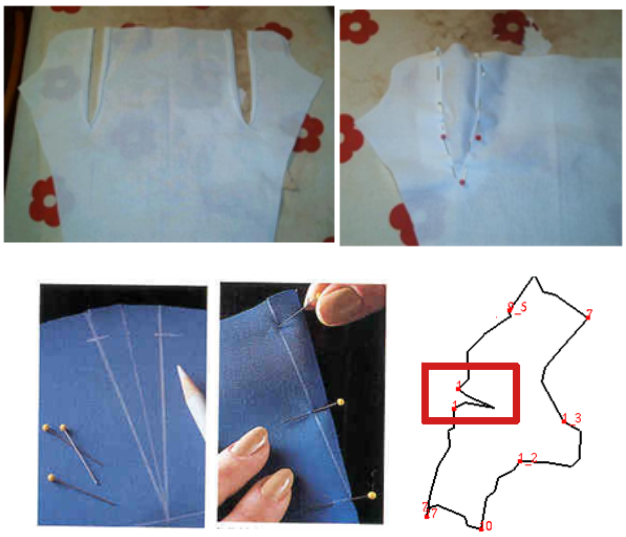


Figure 5.5: Examples of darts and gussets and a dart created by the D-Charts algorithm.

regions cut off, and when possible, reattached to an adjacent chart (Section 5.3).

Lastly, to parameterize surfaces such as cylinders isometrically onto the plane, cuts connecting the chart boundary loops are introduced. At the end of this process a mesh segmentation into quasi-developable charts, which can be used both for texture atlas generation and for model fabrication is produced. Figure 5.3(g) shows the added seems between charts, as well as darts on the ears and each of the feet.

5.3 Pattern design

The following section describes the use of the segmentation algorithm in the design of patterns for stuffed toys and paper models. The algorithm begins by segmenting the model as previously described.

When wrapping the fabric around soft stuffing, such as in soft toys, shearing leads to visible wrinkles. Therefore free-boundary, conformal (low-shear) parameterization is used to unfold the charts in 2D. In all the examples [38] was used.

After the parameterization the algorithm optimizes the charts for sewing needs. First, it tests for overlaps, removes those, and generates gussets. It then shortens ex-

cessively long darts and simplifies chart boundaries. Finally, it packs the charts into a square domain using the method of Levy et al. [22] and prints them including reference points.

Next, the chart optimization steps are described in more detail.

Fixing gussets and overlaps: The parameterization obtained may contain two types of overlaps: gussets (i.e., local overlaps) and regular, global overlaps. The algorithm detects both and cuts the charts to remove them using the method of [22]. The resulting small charts are merged with neighbors if the merging criterion (Section 5.2.3) is satisfied and the merge itself does not create new overlaps.

Shortening darts: Since dart creation is based on a local fitting measure, at times more darts are created than are necessary. The parameterization in practice eliminates such darts, stitching boundary vertices where the sum of angles around the vertex is close to 2π. Such darts are easily identified and then shortened by "gluing" the edges together.

Simplifying boundaries: When creating sewing patterns the boundaries should be as simple as possible. Hence the boundaries between pairs of charts in 2D are simplified using a vertex collapse method. The cost of collapsing a vertex is defined as the sum of the areas of the two triangles formed by the vertex and its neighbors on the boundary (a single triangle is formed on each side of the boundary). The costs are accumulated by adding half the cost of each collapsed vertex to each of its neighbors.

Sewing notations: To ease the sewing, reference points are added to each chart. These help identify corresponding boundary points on adjacent charts and greatly simplify the alignment of the sewing patterns (Figure 5.6).

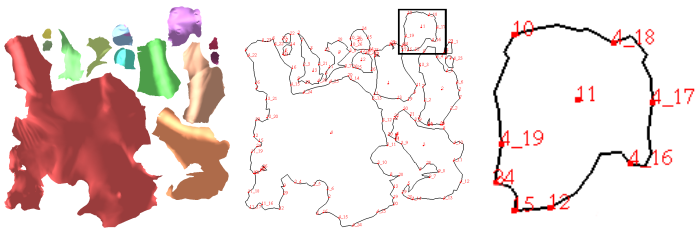


Figure 5.6: Atlas and reference points of bull model.

5.4 D-Charts results

5.4.1 Texture atlases

To evaluate the segmentation algorithm, it was used to generate texture atlases for several models (Figures 5.7 and 5.8). For the parameterization stage the free-boundary stretch minimizing parameterization was used [35]. To evaluate the parameterization the $L_2^{Stretch}$, $L_\infty^{Stretch}$, and L_2^{Shear} metrics were used (Section 2.5).

The parameters used to generate each model and the results obtained are summarized in Table 5.1. All the results were generated on a P4 3GHz machine using the Graphite software package [10]. The runtime is a function of both model size and complexity. For mechanical models such as the Fandisk (10K faces) the time is less than 10 seconds. For free-form models the algorithm takes from 100 seconds to segment a medium-sized 20K faces model (gargoyle) and up to 500 seconds to segment the larger and more complex ones (bunny or feline). Most of the time is spent performing the Lloyd iterations; the hole-filling takes a negligible amount of time and the post-processing takes about 10% of the total time. Our times are comparable to those of the MCGIM algorithm [34]. Note that the method performs well on models of high genus such as the feline, as well as models with boundaries, such as the bunny. The D-Charts algorithm is particularly suitable for segmenting mechanical models (Figure 5.8) for which it accurately captures the planar, cylindrical, and conic parts.

Following is a comparison between the D-Chart algorithm results and those provided by the authors of Iso-charts [50] and by the authors of MCGIM [34]. The D-Charts segmentation method significantly improves all three error metrics for all the models, using the same or a smaller number of charts. The improvement is most significant on mechanical models such as the Fandisk, where D-Charts achieves the minimal $L_2^{Stretch}$ error of one. This indicates that the charts generated are much closer to being developable than those generated using previous methods. Since there is a trade off between developability and compactness, as expected, D-Charts generates charts with slightly longer boundaries. The difference in length is on average about 10% to 15%.

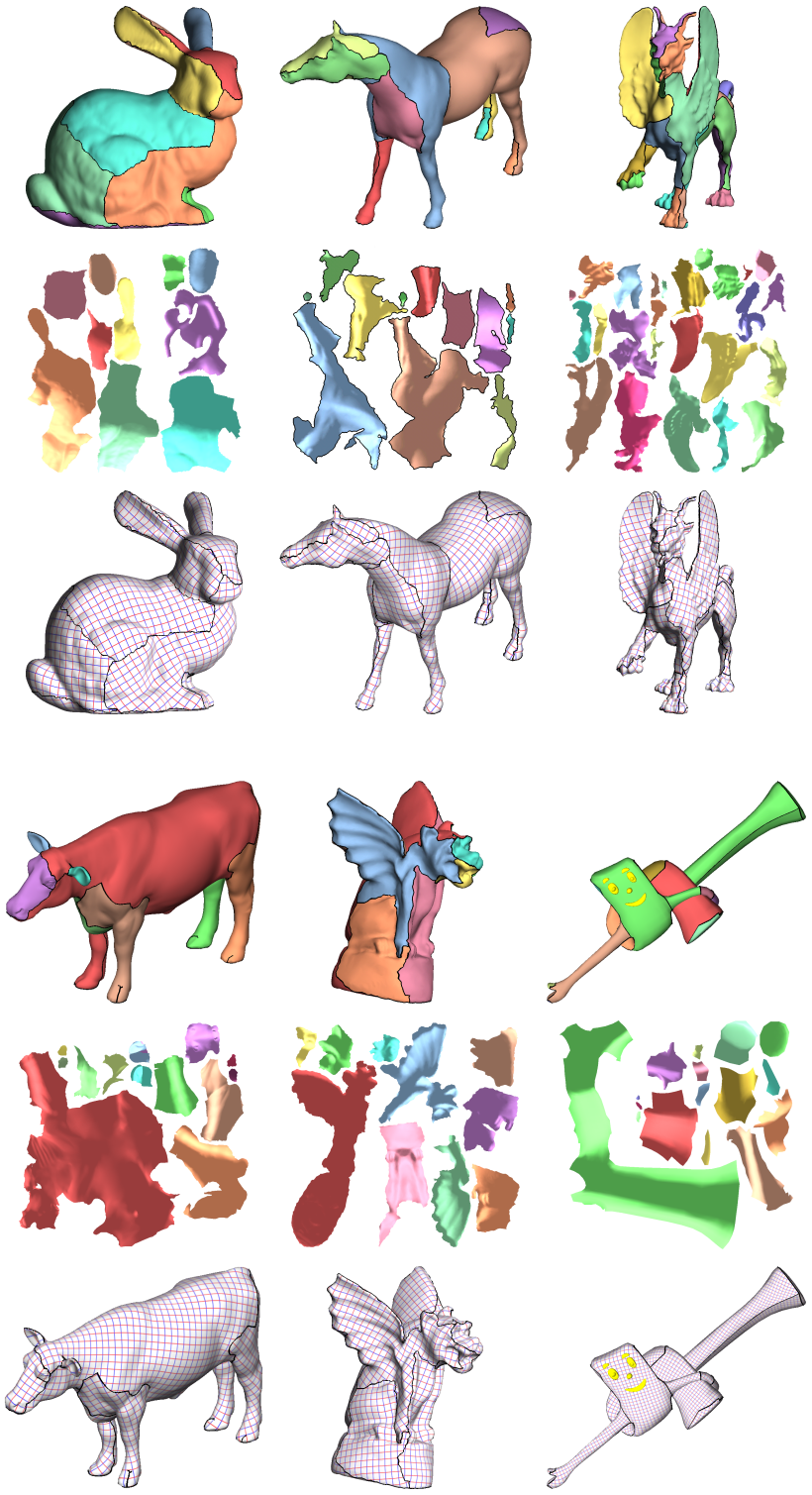


Figure 5.7: Model segmentation and atlas generation. Top rows — mesh segmentation. Middle rows — texture atlases. Bottom rows — parameterization (iso-lines). The statistics for most of the models are given in Table 5.1.

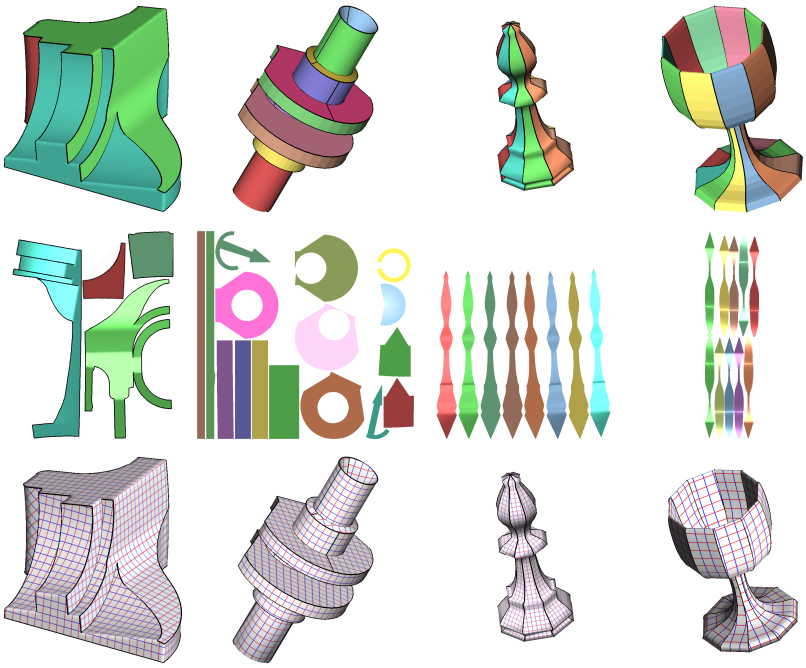


Figure 5.8: Segmentation of mechanical models. First and last rows — mesh segmentation. Second row — texture atlases. Third row - parameterization (iso-lines). The $L_2^{Stretch}$ error for all the textured models is less than 1.00075 (it is not 1 as some models contain small blends or fillets).

5.4.2 Soft toys

Figures 5.9 and 5.10 show sewing examples and the resulting soft-toy models created from the D-Charts sewing patterns. For mechanical models, such as the Fandisk and the bishop, the segmentation results in developable charts. Thus it is possible to reproduce those models accurately from paper (Figure 5.11) or from any other sheet material. For the toy models, distortion must inevitably be introduced. Nevertheless, the generated stuffed toy models capture the shape of the objects. Predictably, minor details are lost, mostly due to fabric resilience.

Figure 5.9: Creating stuffed toys using D-Charts generated patterns.

	Bunny	Horse	Feline	Gargoyle	Fandisk
#faces	69450	19996	41262	20000	9926
D-Charts					
Input #charts	12	15	25	12	6
Total time (sec.)	468	130	482	91	9
Lloyd (sec.)	417	115	446	73	3
Post (sec.)	51	15	36	17	6
Final #charts	10	12	25	10	4
$L_2^{Stretch}$	1.004	1.01	1.01	1.006	1
$L_\infty^{Stretch}$	1.429	2.315	3.488	1.645	1.017
L_2^{shear}	0.006	0.001	0.012	0.008	0
Iso-charts					
#charts	16	13	26	11	4
$L_2^{Stretch}$	1.023	1.035	1.052	1.019	1.021
$L_\infty^{Stretch}$	2.831	2.766	3.401	2.153	2.272
L_2^{shear}	0.021	0.038	0.056	0.022	0.018
MCGIM					
#charts		15	25	10	5
$L_2^{Stretch}$		1.014	1.018	1.009	1.008
$L_\infty^{Stretch}$		2.803	3.563	2.221	2.092
L_2^{shear}		0.014	0.019	0.011	0.012

Table 5.1: Segmentation statistics. For all the models $F_{max} = 0.2$ was used. When available the statistics for models segmented with MCGIM and Iso-charts are provided.

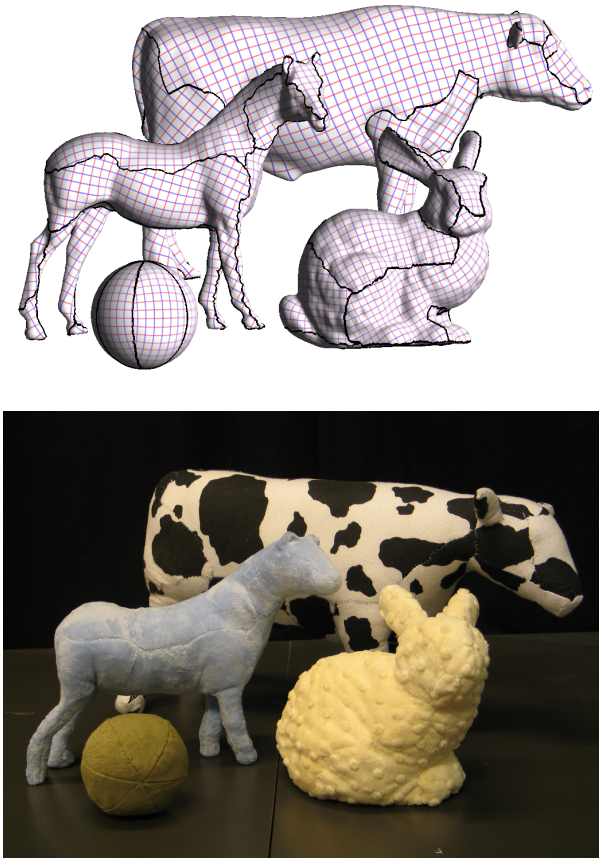


Figure 5.10: Texture atlases and stuffed toys generated using D-Charts.

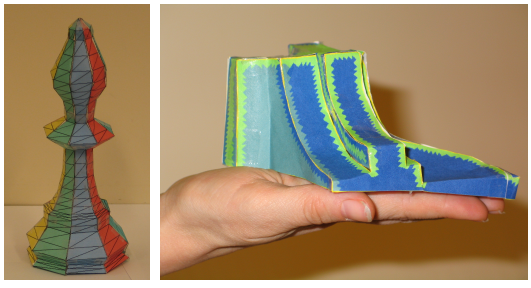


Figure 5.11: Papercraft bishop and Fandisk.

Chapter 6

Developable approximation[1]

This chapter describes the *DCS approximation* algorithm for developable mesh approximation [1] [5]. As noted, the goal of the algorithm is to increase the surface developability while preserving the surface geometry as much as possible. Section 6.1 provides an overview of the method, followed by Section 6.2 which covers each of the algorithm stages in more detail. As explained in Section 1.2 developable surfaces play a key role in both real and virtual fashion design. Therefore, the DCS approximation method was developed in the context of a virtual garment design system. Section 6.3 presents the results of applying the DCS approximation method to garment panels in the virtual design setting.

6.1 DCS approximation method overview

The DCS approximation algorithm modifies surfaces by increasing their developability. In our context, it is assumed that all the seams are specified by the user, thus the algorithm does not introduce additional ones. Hence, the goal is to take each surface panel bounded by the seams and make it developable with as little modification as possible. The $L_2^{Stretch}$, $L_\infty^{Stretch}$, and L_2^{Shear} distortion metrics described in 2.5 are used to quantify the improvement in developability.

To make the panels more developable, the algorithm uses an approach similar to that of the moving-least-squares (MLS) approximation [1, 19, 21]. Working in the projective geometry setting each triangle is processed in turn. First the normal maps

[1] A version of this chapter has been submitted for publication. Decaudin, P., Julius, D., Wither, J., Boissieux, L., Sheffer, A. and Cani, M.P. (2006). Virtual garments: A fully geometric approach for clothing design.

are used to find a locally best fitting developable surfaces for each triangle. Next, each triangle is moved to its corresponding developable surface in Euclidean space. Similar to MLS, local approximation provides a global solution, making the modified surface more developable.

As mentioned in Section 2.3, the normal map of any sufficiently smooth developable surface is a union of curves on the unit sphere. Therefore, the normal map of any sufficiently small region on such surface can be accurately approximated by an arc or a point. The family of surfaces whose normal map is an arc or a point are known as developable surfaces of constant slope [33]. Following the observation above, any sufficiently smooth developable surface can be approximated locally by a developable surface of constant slope.

The algorithm uses the following procedure to perform the local approximation.

- For each triangle on the input mesh it computes a local neighborhood and finds a proxy that best approximates this neighborhood (Section 6.2.1).
- Next, an optimal transformation that moves the triangle to the approximating proxy with minimal change of its vertex positions is computed (Section 6.2.2). This computation is performed independently for each triangle. After the transformation the triangles are no longer connected.
- Finally, to reconnect the mesh the algorithm glues the triangles together, while trying to preserve the newly computed normals and positions (Section 6.2.3). The gluing is applied simultaneously to all the surface panels as they must remain connected.

As shown in Figure 6.1 the footprint of the normal map on the sphere shrinks, making it nearly one dimensional. The process can be repeated to further improve developability. Note that as expected, the improvement in developability comes at the expense of higher deviation from the original surface, thus it is up to user to decide on the right trade off.

Planar parameterization is performed to obtain the actual 2D patterns for the surface panels (Section 6.2.4). The following section describes the algorithm stages in detail.

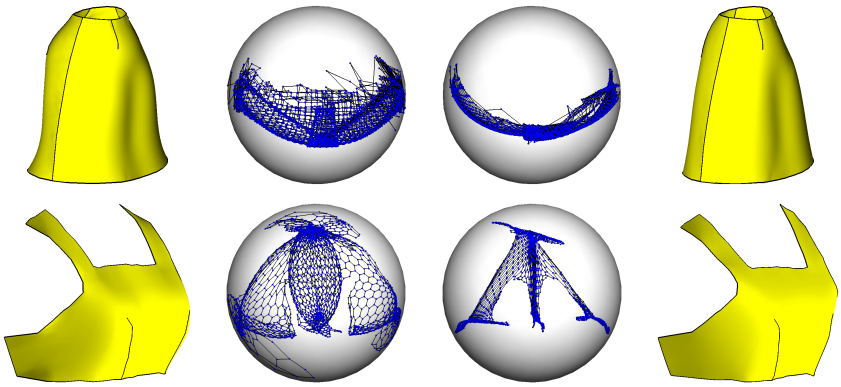


Figure 6.1: (Left) Input to our approximation. The normal map shows the triangle normals and the dual graph edges between the triangles. (Right) Approximation output, the normal map is closer to being one dimensional

6.2 Algorithm stages

6.2.1 Local approximation

The algorithm begins by locally approximating a neighborhood around each mesh triangle by a DCS surface. The neighborhood of a triangle consists of several rings of triangles surrounding it (Figure 6.2).

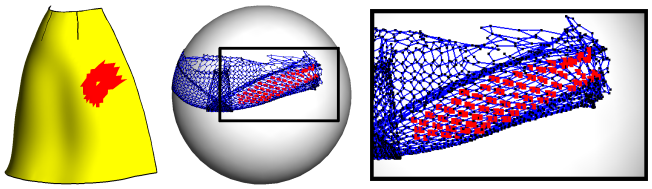


Figure 6.2: Normals in local neighborhood closely fit an arc on the unit sphere.

The approximation is done in terms of the normal map, as it controls the developability of our surface. The results are then translated into Euclidean space when performing the actual triangle transformation. As explained in Section 4.3 a DCS surface can be described by an axis vector N and an angle θ, such that at each point on the surface the angle between the normal to the surface n at that point and the axis is equal to θ. The algorithm computes the axis and the angle by solving the following

constrained optimization problem,

$$\min_{N,\theta} \sum_j ((n_j \cdot N - \cos\theta)/2)^2 \quad subj.\,to \;\; \|N\| = 1 \tag{6.1}$$

where n_j are the normals to the faces in the neighborhood. This formulation is equivalent to that of Eq. 5.4 and is solved in the same manner (Section 5.2.1).

Note that for planes, this definition is not unique. For instance, a plane $P(x) := P_N \cdot x + d = 0$ can be represented by an infinite number of proxies: $(N,\theta) = (P_N, 0)$ or $(N,\theta) = (V, \pi/2) \;\; s.t. \;\; V \perp P_N$. Thus an explicit test if the local neighborhood can be approximated by a plane is performed by finding the axis N that approximates the neighborhood best for $\theta = 0$. If the total fitting error for all triangles in the neighborhood is below a given threshold, the planar proxy is used.

The formulation in Equation 6.1 supports surface approximation with cones and does not distinguish between cones whose apexes lie on the surface and those whose apexes lie outside the approximated surface. The apex of a cone is clearly not developable. Thus, to develop a surface containing an apex and extra dart connecting the apex to the boundary is required. This case can be detected during parameterization and the darts can be added as described in Section 5.2.3. Since extra darts are undesirable in our setting, it is necessary to avoid creating cones whose apex lies on the surface. To achieve this an inequality constraint is introduced into the minimization, requiring that $\|cos\theta\| \leq 0.5$. The next step after developable proxies are found is to transform each triangle.

6.2.2 Triangle Transformation

The triangle transformation is computed in two steps. First the normal n of the triangle is moved to the arc on the unit sphere defined by N and θ. If $\theta > 0$ the normal vector n is rotated around the vector $n \times N$ such that, after rotation $n \cdot N - cos\theta = 0$, otherwise the normal vector is simply set to $n = N$. Next, new positions for the triangles vertices v_1, v_2, v_3, are computed. These positions should be as close as possible to the original positions v_1^0, v_2^0, v_3^0, and the new triangle normal should be equal to n. The new positions

are computed as the solution of a simple quadratic minimization problem,

$$\min_{v_i,d} \sum_{i=1}^{3} (v_i^0 - v_i)^2 \quad subj.to \;\; v_i \cdot n - d = 0 \tag{6.2}$$

It is often useful to fix the position of some of the vertices of the input model. For instance, when developing the skirt mesh (Figure 6.1), to keep the waist-line of a skirt unchanged the positions of the vertices on the waist boundary were fixed. This requirement is easily satisfied by removing the specified vertices from the minimization formula. The system will have a solution as long as one vertex in the triangle remains free.

After all the triangles are moved to their respective local approximating surfaces, the triangles are no longer connected (Figure 6.3), and need to be glued back together as described in the following section.

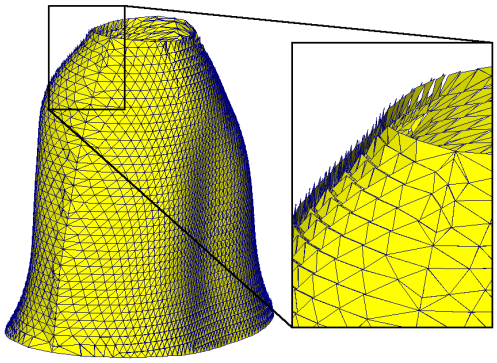


Figure 6.3: Each triangles is translated separately to the approximating developable surfaces. The mesh connectivity is broken.

6.2.3 Gluing

The triangles are glued together by applying an appropriate linear transformation to each triangle. The transformations are formulated using the local frames of the triangles. The local frame is defined using the three triangle vertex positions v_1, v_2 and v_3, and a fourth point v_4 found by offsetting one of the vertices by the triangle normal [42]. The local coordinate frame V is then defined as $V = (v_4 - v_1, v_4 - v_2, v_4 - v_3)$.

The transformation gradient expressed in terms of the local frames before and after the transformation (V and $\tilde{V}$) is $\tilde{V}V^{-1}$ [42]. To preserve the normal and remain close to the original surface, the transformation gradient should be close to identity,

$$\min_{\tilde{v}} \sum_{j} \|\tilde{V}_j V_j^{-1} - I\|_F^2, \tag{6.3}$$

where I is a 3×3 identity matrix, and the index j goes over the mesh triangles. In cases where the surfaces was pre-segmented into a number of panels, this stages is applied to the entire mesh. This is important since we not only want the triangles within each panel to be glued together, but also the panels to remain connected across seams.

6.2.4 Unfolding

To obtain planar patterns, the algorithm unfolds the surface panels (Figure 6.4). ABF++ [38] conformal (shear minimizing) parameterization is used to unfold the panels. ABF++ generates zero-distortion parameterization for truly developable surfaces, and minimizes shear when distortion is inevitable. Figure 6.5 shows two outfits with their corresponding texture maps and the resulting 2D patterns.

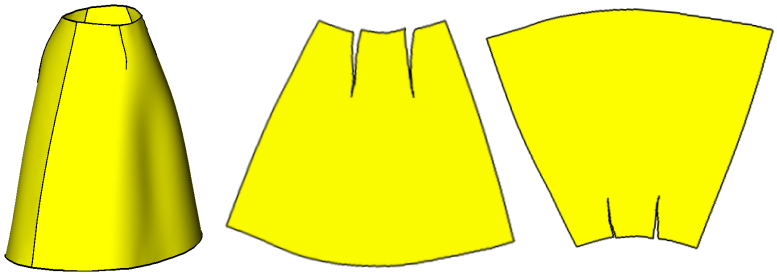


Figure 6.4: Panel unfolding using ABF++[38].

6.3 DCS approximation results

The DCS approximation algorithm was developed in the context of a virtual garment design system. Input meshes were generated using an intuitive sketching system based

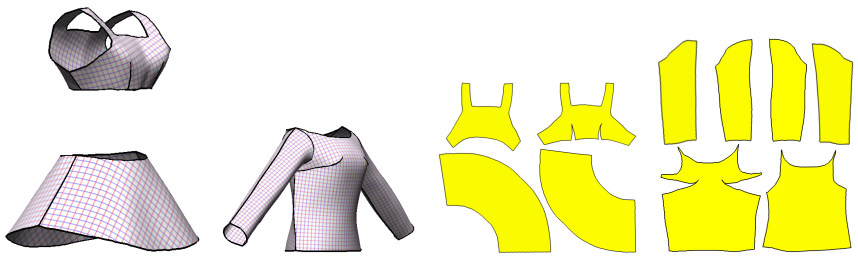


Figure 6.5: Texture mapped garments and corresponding patterns.

on that of Turquin et al. [43]. A 3D shape for the virtual garment is deduced from contour lines drawn by the user in 2D over a front (or back) view of a virtual mannequin. The sketching system was further enhanced to enable the addition of seam-lines and darts [5]. Figure 6.6(left) shows an example sketch.

Figure 6.6: From sketch to garment. Left to right: sketched contours and darts; 3D shape computed using distance field; piecewise developable surface; final virtual garment, compared with the real one sewn from the 2D patterns we output.

Next, each surface panel was approximated with a developable surface and unfolded into a 2D pattern using the DCS approximation method described above (Figure 6.6(center)). All the results were generated on a P4 3GHz machine using the Graphite software package [10]. The runtime of the approximation method is a function of the mesh size, where one iteration of approximation on the 7K faces skirt (Figure 6.7) takes about 29 seconds.

The developed surfaces and the 2D patterns were then used as input to a procedural fold modeling algorithm, in order to create realistic looking 3D buckling folds without performing cloth simulation [5]. For the shirt and skirt models, the 2D patterns were also used to sew real garments as shown in Figures 6.6(right) and 6.7(right).

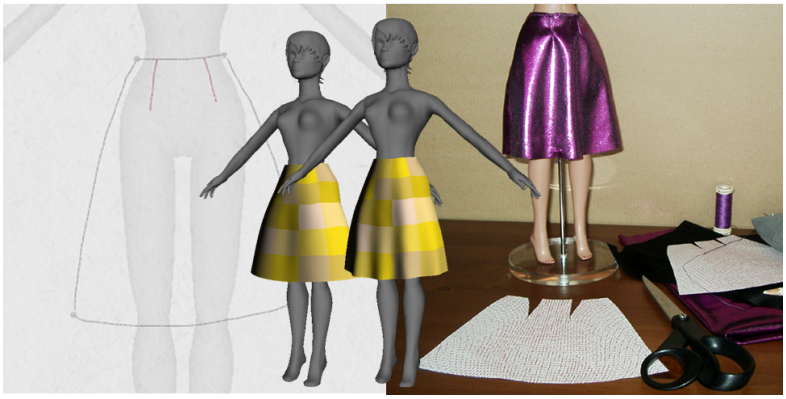


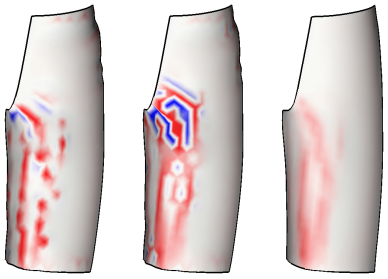
Figure 6.7: From sketch to garment. Left to right: sketched contours and darts; piecewise developable surface; final virtual garment, compared with the real one sewn from the 2D patterns.

To evaluate the improvement in the degree of developability for each surface, we measured the distortion created during the unfolding process. Table 6.1 lists the statistics for the models shown in the paper. To test the method on an example with higher initial distortion, the skirt model without any darts was used as input. As shown in the table, the distortion before the approximation is quite high, but is drastically reduced after applying our approximation algorithm.

Additionally, we compared the DCS approximation method with the Gaussian curvature based method of Wang and Tang [44], which is currently the only other method applicable to 3D meshes. Table 6.1 shows that the improvement achieved by DCS approximation is better. Moreover, two more significant differences are noted. First, the DCS approximation does not introduce folds and wrinkles into the mesh. This effect is particularly undesirable in our setting, since a well fitting garment should ideally have no wrinkles beyond those caused by external physical forces such as collisions and gravity. Figure 6.8 shows a comparison of the mean curvature across the surface, and demonstrates that DCS approximation actually reduces the mean curvature. Second, the DCS approximation method is not influenced by the existence of local minima encountered when optimizing the Gaussian curvature, and is therefore capable of operating on surfaces which are not initially sufficiently close to being developable.

	skirt	shirt	bra	mini skirt	pant leg*	skit no darts
#faces	6818	7016	1366	2923	936	6818
#panels	2	6	2	2	1	2
input mesh						
$L_2^{stretch}$	1.003	1.005	1.0015	1.0012	1.008	1.01
$L_\infty^{stretch}$	1.122	1.137	1.119	1.0855	1.0786	1.537
L_2^{shear}	4.6e-5	3e-4	1.5e-4	3.5e-5	8.4e-5	1.4e-4
modified mesh						
$L_2^{stretch}$	1.00035	1.0014	1.0005	1.0002	1.0001	1.0005
$L_\infty^{stretch}$	1.056	1.075	1.096	1.021	1.0157	1.064
L_2^{shear}	2.99e-6	6.21e-5	3.2e-5	9.2e-7	2.4e-6	2.9e-6

Table 6.1: DCS approximation statistics. *For comparison, the distortion for the output generated by Wang [44] is ($L_2^{stretch} = 1.00015,\ L_\infty^{stretch} = 1.0232,\ L_2^{shear} = 3.4e-6$).



(a) (b) (c)

Figure 6.8: Curvature comparison with Wang et al. [44]: (a) input mesh; (b) result provided by Wang; (c) DCS approximation output. The coloring shows the mean curvature of the surface. The statistics for the models are listed in Table 6.1.

Chapter 7

Summary and conclusions

This thesis focuses on developable surfaces, 3D meshes, and mesh processing algorithms that take advantage of the developable surface properties. The main challenge faced when dealing with developable surfaces represented as meshes is how to characterize these surfaces. This thesis presents a novel method of capturing these surfaces. The *DCS descriptor* and *fitting error* defined in Section 4.3 provide simple and robust tools for detecting, measuring, and approximating developable charts in meshes. This tool is more advanced compared to previously used techniques such as usage of planar charts, which are developable yet quite restrictive, or Gaussian curvature based methods which are subject to numerical instabilities. Based on this concept two novel algorithms were created. First, an algorithm named *D-Charts* for (nearly) developable mesh segmentation (Chapter 5). Second, *DCS approximation*, an algorithm for developable surface approximation (Chapter 6).

Mesh segmentation has numerous applications in computer graphics [37]. As described in Chapter 1, developable segmentation is the first step required when creating model atlases for texturing or patterns. To preserve the texture and other attributes during mapping, the parameterizations should be as close as possible to being isometric, since only these do not introduce distortion. This is possible only if the charts are nearly developable.

As demonstrated by the examples in Section 5.4, D-Charts successfully segments meshes into compact (nearly) developable charts. Mechanical models are segmented into their developable components, producing isometric texture atlases, while for other models for which a compact developable segmentation does not exist, the segmentation is based on a user-prescribed developability tolerance. As evident from the comparison

in table 5.1 the algorithm is more suitable for texture atlas generation than previous segmentation techniques. The D-Charts algorithm also provides a first step toward fully automatic pattern design, an important problem which previous research did not seriously address. Based on the segmentation algorithm an automatic method for pattern design, focusing on the design of soft, stuffed toys is introduced. To demonstrate the practicality of the technique, design patterns for several popular computer graphics models were created and then used to construct physical copies of the models out of fabric or paper.

Approximating surfaces with developable surfaces is a useful feature in many design processes. Chapter 1 explains how developable surfaces play a key role in any manufacturing industry requiring patterns for sewing, metal forming and forging, or any other fabrication applications where 3D objects are constructed from sheets of material.

Chapter 6 describes the developable approximation method developed as part of this thesis. As demonstrated in Section 6.3, DCS approximation method is particularly useful, since it may be applied to meshes that are not close to being developable. Moreover, in contrast to previous methods, it increases surface developability without introducing additional wrinkles. The method was combined with a sketching interface and fold modeling system to provide a novel solution for the design of virtual garments. In this system, an initial rough geometry of the garment, created using the sketching interface, is approximated by a piecewise developable surface. Since the approximated surface is piecewise developable, computing the 2D sewing patterns and distortion-free texture mapping becomes straight-forward, and real replicas of the designed garments can be sewn.

Together these algorithms demonstrate advantages of using the *DCS descriptor* and *fitting error* for detecting, measuring, and approximating developable charts in meshes.

Chapter 8

Future work

An important consideration when modeling 3D objects is symmetry. For instance, most garment patterns and many soft toys have a clear symmetry axis. An important improvement to the D-Charts algorithm, is to correctly identify this symmetry before processing the models in order to enforce the desired symmetry in the 2D patterns. Moreover, for any machine based fabric cutting or stamping method, if several charts have the same shape (up to mirroring), the cutting becomes much simpler.

Common to all *K-means* clustering or Max-Lloyd methods is the manual selection of the parameter *K*. While D-Charts does partially address this problem using *holes* and *merging* (Sections 5.2.2 and 5.2.3), an initial reasonable estimate is still required. A generic method to provide an initial guess applicable to any type of model would further improve the algorithm.

The shape of chart boundary curves is also of great importance in many applications. Currently the DCS approximation allows only simple constraints of fixing vertices in place, while the D-Charts algorithm does not address this issue at all. Smooth boundaries and curvature constraints would probably simplify the cutting and sewing processes.

Section 6.2.1 of the DCS approximation algorithm, which explains how local neighborhoods are found, raises an interesting question — How do we estimate the optimal neighborhood size? While currently this is left as a user defined parameter, it would be interesting to find optimal neighborhoods automatically. One possible approach would be to use a grow-reseed method, such that neighborhoods are grown and shrunk until they are optimally fitted around each triangle.

Finally, it would be of great interest to further broaden the set of developable sur-

faces handled. By extending the proxy definitions to capture more general developable surfaces, or improving the methods to better handle non-smooth surfaces, better and less restrictive results may be achieved.

Bibliography

[1] Marc Alexa, Johannes Behr, Daniel Cohen-Or, Shachar Fleishman, David Levin, and Claudio T. Silva. Point set surfaces. In *VIS '01: Proceedings of the conference on Visualization '01*, pages 21–28, Washington, DC, USA, 2001. IEEE Computer Society.

[2] H.-Y. Chen, I.-K. Lee, Stefan Leopoldseder, Helmut Pottmann, Thomas Randrup, and Johannes Wallner. On surface approximation using developable surfaces. *Graphical Models and Image Processing*, 61(2):110–124, 1999.

[3] Paolo Cignoni, Claudio Rocchini, and Roberto Scopigno. Metro: measuring error on simplified surfaces. Technical report, Paris, France, France, 1996.

[4] David Cohen-Steiner, Pierre Alliez, and Mathieu Desbrun. Variational shape approximation. *ACM Trans. Graph.*, 23(3):905–914, 2004.

[5] Philippe Decaudin, Dan Julius, Jamie Wither, Laurence Boissieux, Alla Sheffer, and Marie-Paule Cani. Virtual garments: A fully geometric approach for clothing design. *Submitted*, 2006.

[6] Gershon Elber. Model fabrication using surface layout projection. *Computer-Aided Design*, 27(4):283–291, 1995.

[7] M. S. Floater and K. Hormann. Surface parameterization: a tutorial and survey. In N. A. Dodgson, M. S. Floater, and M. A. Sabin, editors, *Advances in Multiresolution for Geometric Modelling*, Mathematics and Visualization, pages 157–186. Springer, Berlin, Heidelberg, 2005.

[8] Michael Garland, Andrew Willmott, and Paul S. Heckbert. Hierarchical face clustering on polygonal surfaces. In *SI3D '01: Proceedings of the 2001 symposium on Interactive 3D graphics*, pages 49–58. ACM Press, 2001.

[9] Natasha Gelfand and Leonidas J. Guibas. Shape segmentation using local slippage analysis. In *SGP '04: Proceedings of the 2004 Eurographics/ACM SIGGRAPH symposium on Geometry processing*, pages 214–223, New York, NY, USA, 2004. ACM Press.

[10] Graphite, 2003. http://www.loria.fr/∼levy/Graphite/index.html.

[11] Xianfeng Gu, Steven J. Gortler, and Hugues Hoppe. Geometry images. In *SIGGRAPH '02: Proceedings of the 29th annual conference on Computer graphics and interactive techniques*, pages 355–361. ACM Press, 2002.

[12] M. Guthe and R. Klein. Automatic texture atlas generation from trimmed NURBS models. *Computer Graphics Forum*, 22(3):453–453, 2003.

[13] Takeo Igarashi and Dennis Cosgrove. Adaptive unwrapping for interactive texture painting. In *SI3D '01: Proceedings of the 2001 symposium on Interactive 3D graphics*, pages 209–216. ACM Press, 2001.

[14] Dan Julius, Vladislav Kraevoy, and Alla Sheffer. D-charts: Quasi-developable mesh segmentation. In *Computer Graphics Forum, Proceedings of Eurographics 2005*, volume 24, pages 581–590, Dublin, Ireland, 2005. Eurographics, Blackwell.

[15] Sagi Katz, George Leifman, and Ayellet Tal. Mesh segmentation using feature point and core extraction. *The Visual Computer*, 21(8-10):649–658, 2005.

[16] Sagi Katz and Ayellet Tal. Hierarchical mesh decomposition using fuzzy clustering and cuts. *ACM Trans. Graph.*, 22(3):954–961, 2003.

[17] Simon Kolmanic and Nikola Guid. A new approach in cad system for designing shoes. *Journal of Computing and Information Technology*, 11(4), 2003.

[18] Venkat Krishnamurthy and Marc Levoy. Fitting smooth surfaces to dense polygon meshes. In *SIGGRAPH '96: Proceedings of the 23rd annual conference on Computer graphics and interactive techniques*, pages 313–324. ACM Press, 1996.

[19] In-Kwon Lee. Curve reconstruction from unorganized points. *Comput. Aided Geom. Des.*, 17(2):161–177, 2000.

[20] Stefan Leopoldseder and Helmut Pottmann. Approximation of developable surfaces with cone spline surfaces. *Computer-aided Design*, 30(7):571–582, 1998.

[21] D. Levin. The approximation power of moving least-squares. *Math. Comput.*, 67(224):1517–1531, 1998.

[22] Bruno Lévy, Sylvain Petitjean, Nicolas Ray, and Jérome Maillot. Least squares conformal maps for automatic texture atlas generation. In *SIGGRAPH '02: Proceedings of the 29th annual conference on Computer graphics and interactive techniques*, pages 362–371. ACM Press, 2002.

[23] Stuart P. Lloyd. Least squares quantization in PCM. *Bell Telephone Laboratory Memorandum (1957), reprint IEEE Transactions on Information Theory IT-28, 2 (Mar 1982), 129-I37*, 1957.

[24] Jérôme Maillot, Hussein Yahia, and Anne Verroust. Interactive texture mapping. In *SIGGRAPH '93: Proceedings of the 20th annual conference on Computer graphics and interactive techniques*, pages 27–34, New York, NY, USA, 1993. ACM Press.

[25] Maya cloth. In *Maya cloth user manual, Alias*, 2004.

[26] J. McCartney, B. K. Hinds, and B. L. Seow. The flattening of triangulated surfaces incorporating darts and gussets. *Computer-Aided Design*, 31(4):249–260, 1999.

[27] Jun Mitani and Hiromasa Suzuki. Making papercraft toys from meshes using strip-based approximate unfolding. *ACM Trans. Graph.*, 23(3):259–263, 2004.

[28] Barbara Pearl. *Math in Motion: Origami in the Classroom K-8*. Math in Motion, 1999.

[29] M. Peternell. Recognition and reconstruction of developable surfaces from point clouds. In *Proceedings of 'Geometric Modeling and Processing 2004'*, pages 301–310. Elsevier Science, 2004.

[30] Martin Peternell. Developable surface fitting to point clouds. In *Computer Aided Geometric Design*, pages 785–803, 2004.

[31] H. Pottmann and J. Wallner. Approximation algorithms for developable surfaces. *Comput. Aided Geom. Design*, 16:539–556, 1999.

[32] Helmut Pottmann and T. Randrup. Rotational and helical surface approximation for reverse engineering. *Computing*, 60(4):307–322, 1998.

[33] Helmut Pottmann and Johannes Wallner. *Computational Line Geometry*. Springer Verlag, 2001.

[34] P. V. Sander, Z. J. Wood, S. J. Gortler, J. Snyder, and H. Hoppe. Multi-chart geometry images. In *SGP '03: Proceedings of the Eurographics/ACM SIGGRAPH symposium on Geometry processing*, pages 146–155. Eurographics Association, 2003.

[35] Pedro V. Sander, Steven J. Gortler, John Snyder, and Hugues Hoppe. Signal-specialized parametrization. In *EGRW '02: Proceedings of the 13th Eurographics workshop on Rendering*, pages 87–98. Eurographics Association, 2002.

[36] Pedro V. Sander, John Snyder, Steven J. Gortler, and Hugues Hoppe. Texture mapping progressive meshes. In *SIGGRAPH '01: Proceedings of the 28th annual conference on Computer graphics and interactive techniques*, pages 409–416. ACM Press, 2001.

[37] Ariel Shamir. A formulation of boundary mesh segmentation. In *3DPVT*, pages 82–89, 2004.

[38] A. Sheffer, B. Lévy, M. Mogilnitsky, and A. Bogomyakov. ABF++: fast and robust angle based flattening. *ACM Trans. Graph.*, 24(2):311–330, 2005.

[39] Alla Sheffer and John C. Hart. Seamster: inconspicuous low-distortion texture seam layout. In *VIS '02: Proceedings of the conference on Visualization '02*, pages 291–298. IEEE Computer Society, 2002.

[40] Dennis R. Shelden. Digital surface representation and the constructibility of gehry's architecture, 2002.

[41] Olga Sorkine, Daniel Cohen-Or, Rony Goldenthal, and Dani Lischinski. Bounded-distortion piecewise mesh parameterization. In *VIS '02: Proceedings of the conference on Visualization '02*. IEEE Computer Society, 2002.

[42] R. W. Sumner and J. Popovic. Deformation transfer for triangle meshes. *ACM Trans. Graph.*, 23(3):399–405, 2004.

[43] E. Turquin, M-P. Cani, and J. F. Hughes. Sketching garments for virtual characters. *EG Workshop on Sketch-Based Interfaces and Modeling*, August 2004.

[44] C. C. L. Wang and K. Tang. Achieving developability of a polygonal surface by minimum deformation: a study of global and local optimization approaches. *The Visual Computer*, 20(8-9):521–539, 2004.

[45] C. C. L. Wang, Y. Wang, and M. M. Yuen. On increasing the developability of a trimmed nurbs surface. *Eng. Comput. (Lond.)*, 20(1):54–64, 2004.

[46] Charlie C. L. Wang, Kai Tang, and Benjamin M. L. Yeung. Freeform surface flattening based on fitting a woven mesh model. *Computer-Aided Design*, 37(8):799–814, 2005.

[47] Jianhua Wu and Leif Kobbelt. Structure recovery via hybrid variational surface approximation. In *Computer Graphics Forum, Proceedings of Eurographics 2005*, volume 24, page 277, Dublin, Ireland, 2005. Eurographics, Blackwell.

[48] Xiang, Held, and Mitchell. Fast and effective stripification of polygonal surface models (short). In *SODA: ACM-SIAM Symposium on Discrete Algorithms (A*

Conference on Theoretical and Experimental Analysis of Discrete Algorithms), 1999.

[49] Hitoshi Yamauchi, Stefan Gumhold, Rhaleb Zayer, and Hans-Peter Seidel. Mesh segmentation driven by gaussian curvature. *The Visual Computer*, 21(8-10):649–658, September 2005.

[50] Kun Zhou, John Synder, Baining Guo, and Heung-Yeung Shum. Iso-charts: Stretch-driven mesh parameterization using spectral analysis. In *Proceedings of the Eurographics/ACM SIGGRAPH symposium on Geometry processing*. Eurographics Association, 2004.

[51] Inc Ziena Optimization. http://www.ziena.com/knitro.html.