Pacific Graphics 2009
S. Lee, D. Lischinski, and Y. Yu
(Guest Editors)

Volume 28 (2009), Number 7

Interactive Cover Design Considering Physical Constraints

Yuki Igarashi[1],Takeo Igarashi[1,2] and Hiromasa Suzuki[1]

[1]The University of Tokyo, Japan
[2]JST ERATO

Abstract

We developed an interactive system to design a customized cover for a given three-dimensional (3D) object such as a camera, teapot, or car. The system first computes the convex hull of the input geometry. The user segments it into several cloth patches by drawing on the 3D surface. This paper provides two technical contributions. First, it introduces a specialized flattening algorithm for cover patches. It makes each two-dimensional edge in the flattened pattern equal to or longer than the original 3D edge; a smaller patch would fail to cover the object, and a larger patch would result in extra wrinkles. Second, it introduces a mechanism to verify that the user-specified opening would be large enough for the object to be removed. Starting with the initial configuration, the system virtually "pulls" the object out of the cover while avoiding excessive stretching of cloth patches. We used the system to design real covers and confirmed that it functions as intended.

Categories and Subject Descriptors (according to ACM CCS): Computer Graphics [I.3.6]: Methodology and Techniques—Interaction Techniques; Computer Graphics [I.3.5]: Computational Geometry and Object Modeling—Geometric Algorithms

1. Introduction

It is often desirable to have a cover for personal belongings such as cameras, teapots, or cars to protect these items from dust and damage. Covers can also be useful for maintaining the temperature of the contents. However, off-the-shelf cover products are not always available for every object, so it is necessary to develop technology to create customized covers. However, various physical constraints make this kind of technology very difficult for non-experts to use. The design process involves finding two-dimensional (2D) patches sufficiently large to cover the entire surface. The opening must also be large enough to allow the object to be removed easily. A computer program could be of great help in solving these kinds of problems.

This paper presents an interactive system to design a cover for any given three-dimensional (3D) object. Figure 1 shows the overall process of the cover design. The modeling process starts with a 3D model of the target object. The acquisition of this 3D model is in itself a difficult problem, but it is beyond the scope of this paper. The system first computes the convex hull of the target model. The user segments it into several cloth patches by drawing and erasing segmentation boundaries on the 3D surface. The system also provides automatic segmentation. The system then automatically flattens the cloth patches and visualizes the shape of the resulting 3D cover. The user then specifies the cover opening by marking a part of the segmentation boundaries as the opening. The system examines the opening and alerts the user if the opening will be too small to allow for the removal of the object.

This paper makes two technical contributions. First, it introduces a specialized flattening algorithm for cover patches. The system makes every 2D edge in the flattened mesh longer than the original 3D edge, because cloth patches smaller than the original 3D surface will fail to cover the area. A larger patch is less problematic because it will simply result in extra wrinkles. The system begins with a standard flattening result and iteratively refines the result by identifying shortened edges. It runs the optimization again using the increased weights for the shortened edges. As a result, all shortened 2D edges have almost equal in length to the original 3D edges; differences are negligible.

Second, the paper introduces a mechanism to verify that the object can actually be removed through the given opening. Beginning with the initial configuration, the system virtually “pulls” the object out of the cover while avoiding ex-

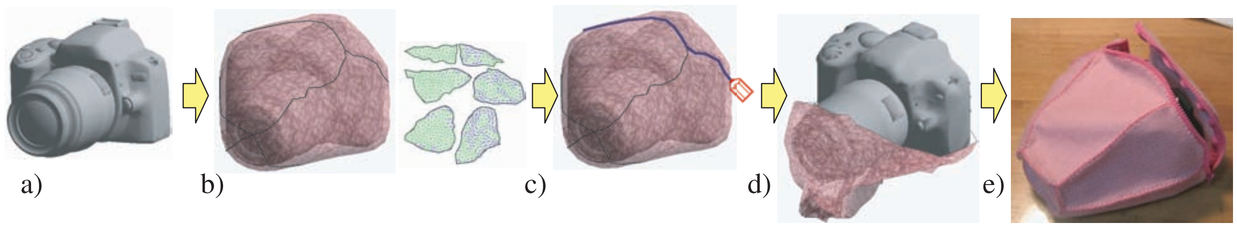


Figure 1: *Overview of our system: (a) input three-dimensional model; (b) segmentation of the convex hull and shrink-free flattening of the patches; (c) specifying an opening; (d) uncovering test; (e) cover created according to the pattern.*

cessive stretching of cloth patches. This is again an iterative process. The system gradually expands the opening to allow the object to pass and pushes the cover along the surface of the object away from the opening. The system also applies a relaxation process [IH02] to prevent excessive stretching of the cloth material. This process terminates when the object is sufficiently uncovered or if the object becomes stuck.

2. Related Work

Researchers have recently proposed some interesting systems to support the computer design of real-world objects. Some systems use a given 3D model as input and generate a physical model as output, e.g., a wet suit [Wan08], a paper craft [MS04], a stuffed animal [JKS05], and a knitted animal [IIS08a]. Some systems allow for the interactive design of a physical model by allowing the user to sketch the object, e.g., a garment [DJW*06], a plush toy [MI07], and a knitted animal [IIS08b]. These systems differ from traditional industrial computer-aided design systems in that they make the design process accessible to non-experts by incorporating the properties of physical materials, such as paper or cloth, into the modeling process. We built on these previous attempts and explored the possibility of using a design process to verify the capability of physical objects.

Within the field of computer graphics, considerable attention has been paid to surface flattening, also called mesh parameterization. Most researchers have focused on texture mapping, with an emphasis on conformality [LSNJ02,SLMB05]. They have also investigated size preservation, e.g., low-stretch parameterization [YBS08], shape-preserving parameterization [LZX*08], and length preservation of feature curves [Wan08]. McCartney et al. presented a method for flattening the candidate triangle sets iteratively while evaluating the local distortion energy of particle movements [MHS99]. Zhong and Xu introduced a flattening method to retain the size and area of the original 3D surface [ZX06]. However, these methods treat shrinkage and stretch equally, and to the best of our knowledge no methods have been developed for shrink-free flattening.

Recent computer-aided process engineering systems such as DELMIA [das] and Tecnomatrix Robcad [rob] have produced various dynamic and static collision check methods. For example, they can simulate robot movement within a car seat; they use 3D models to test all possible trajectories and assess the possibility of collision. Most existing methods focus on collisions between rigid objects [KL00, ZHKM08, ZKVM06]. Our system can handle the complex interactions between rigid and deformable objects, providing an approximate solution sufficient for guiding non-expert users in casual design tasks.

Various methods have been proposed for cloth simulation. Vollino et al. presented techniques for simulating the motion and the deformation of cloth [VCT95]. Bridson et al. presented an efficient and robust collision handling algorithm for the interaction between cloth and rigid objects [BFA02]. Our contribution is orthogonal to these methods in that they simulate the physical behavior of cloth responding to external forces while we discuss how to move the cloth actively to uncover the object. We use relatively simple cloth simulation, but it is possible to combine our system with these more realistic simulation methods.

3. Overview

This section describes the overall process of cover design, mainly from the user's perspective. The details of the main algorithms are described.

3.1. Model Acquisition

The user first needs to obtain a 3D model M of the target object. The model can be created from scratch using a standard modeling software [may, 3ds], or the object can be scanned using a 3D scanner. It might be possible to use a vision-based technique to reconstruct a 3D model. In some cases, 3D models of specific commercial products are already available on the Internet. We have used various combinations of these methods here, but the details of model acquisition are outside the scope of this paper. Our system requires only a polygon soup or point cloud model, because the following computations apply a convex hull.

Figure 2: *Examples of real covers.*

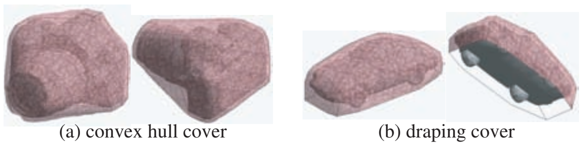


Figure 3: *Computation of cover geometry.*

3.2. Specifying the Cover Geometry

After a 3D model of the target object has been obtained, the next step is to specify the cover geometry. Currently our system supports two kinds of covers: wrapping and draping covers. A wrapping cover completely encloses the target object, e.g., a bottle cover or a protective carrying case for digital equipment (see Figure 2, left). A draping cover only covers the top and sides of the target object, leaving the bottom open, e.g., a piano cover or a tea cozy (see Figure 2, middle and right).

A cover does not need to conform to every detail of the target model. Based on observations of real covers (Figure 2), we decided to use the convex hull of the target model as the cover geometry, as shown in Figure 3a. In the case of a draping cover, the system projects all of the vertices to the ground plane and includes them in the computation of the convex hull. The system then removes all of the faces on the ground plane, as shown in Figure 3b. It is possible to use a more tightly fitted geometry such as an alpha hull [EM94], but we decided to use the simplest method because many real covers apply a convex hull. Convexity also simplifies the design of flattening and uncovering algorithms. Another advantage of convex hull cover is that it uses less cloth material than tighter fitting covers, which makes the physical construction easier.

The system resamples the convex hull mesh $CH(M)$ so that it consists of near-equilateral, uniformly sized triangles [Tur92]. This ensures that the relaxation process will accurately mimic cloth behavior, as shown in Figure 4b. The system uses this resampled convex hull G as the guiding geometry in the relaxation process (see Section 5.1). The system then generates the target cover geometry C_0 by slightly enlarging the resampled convex hull mesh to give it enough play, as shown in Figure 4c. Specifically, the system duplicates the resampled convex hull mesh G and moves each vertex slightly to its normal direction.

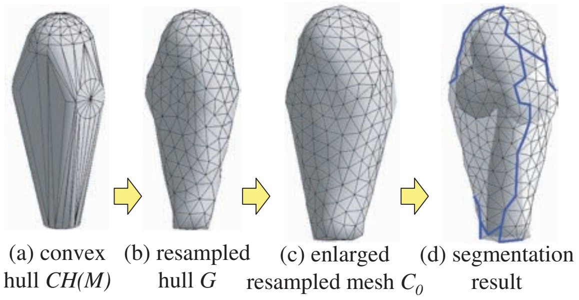


Figure 4: *Resampling and segmentation.*

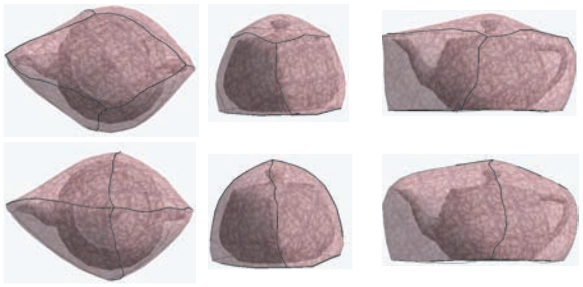


Figure 5: *Preview of the resulting cover shape C.*

3.3. Segmentation

After the target cover geometry C_0 has been obtained, the next task is to segment it into almost planar surface patches. The user segments it into several cloth patches manually. The system also provides an automatic segmentation; if the user can not design.

Various automated segmentation methods have been proposed [JKS05, LA06, LCWK07]. However automatic segmentation does not always return ideal results, partly because the system cannot understand the semantics of the target model. Differing segmentation results in very different final appearances (see Figure 5), so it is important that the user is able to edit the segmentation manually. Our system provides a simple interface for designing segmentation, allowing the user to draw and erase segmentation boundaries manually. Our current implementation places segmentation boundaries only on the mesh edges.

However, the system also provides an automatic segmentation method for novice user. We use a region-growing method [Llo57, CSAD04] for mesh segmentation (Figure 6). The system first randomly distributes a fixed number of proxies on the surface. We use six proxies for a wrapping cover and five for a draping cover. The user can change the number of proxies. The system then partitions the surface via region growing around the proxies and updates the proxies so that they minimize the distortion error. The system repeats partitioning and fitting alternatively until convergence. Since this algorithm starts with random seeds, it generates different

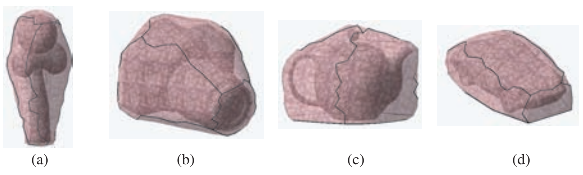


Figure 6: *Example of mesh segmentation: (a, b) convex hull covers; (c, d) draping covers.*

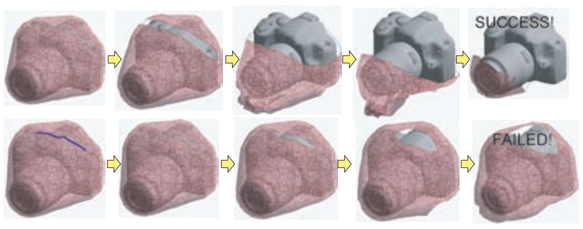


Figure 7: *Uncovering test: success (top); failure (bottom).*

results each time. So the user is encouraged to request segmentation again if the result is not satisfactory. We tested a hierarchical mesh segmentation methods [GWH01, AFS06] on EfPiSoft [efp]. However, the quality of results is similar to the region-growing method [Llo57, CSAD04] because we use a coarse mesh (around 500-2000 vertices). There is an extension to use proxies that represent developable surfaces [JKS05], but we found that simple planar proxies are more suitable for cover design because planar proxies generate more straight boundaries than flexible proxies.

3.4. Flattening

The system flattens the patches of the target cover geometry C_0 after segmentation. Flattening is an automatic process, and the result is shown in a 2D view as a 2D cloth pattern in Figures 1b and c. Section 4 describes the details of the flattening algorithm. The user can manually modify the flattening results by deforming and smoothing patch boundaries in the 2D view. This manual editing is useful for removing undesirable zigzags along the boundary while preserving important features.

The system provides a preview of the resulting cover shape C by applying a relaxation process (Figure 5). It is a simplified cloth simulation [IH02]. It moves the vertices to prevent excessive elongation and shrinkage of edges. It also tries to make the dihedral angles between neighboring faces 180 degrees except around segmentation boundaries. If the results are unsatisfactory, the user can easily change the segmentation by erasing and redrawing the seam lines.

3.5. Specifying the opening of the Cover

The user specifies the cover opening by cutting some of the segmentation boundaries. The system then verifies that the

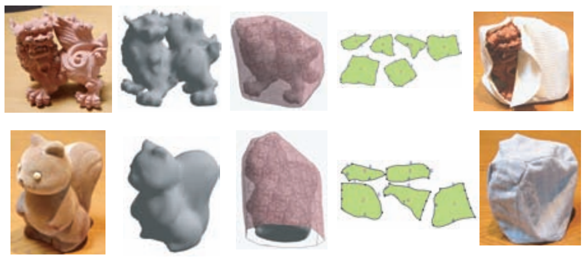


Figure 8: *Examples of the real cover using our system. A convex hull cover of shiisa (upper) and a draping cover of squirrel (bottom).*

object can be removed from the cover by running an uncovering test. If the given opening is too small, the inner object cannot be removed (Figure 7, bottom). In this case, the system alerts the user to design a larger opening (Figure 7, top). Section 5 describes the details of the uncovering test. A draping cover will not require specification of an opening or an uncovering test because the opening has already been defined and it is always possible to remove the cover. Uncovering a draping cover is always possible because the opening is obtained by projecting all vertices of the object and the cover is convex hull of the object.

3.6. Printing and Sewing

At this point, the user can create a real cover by cutting and stitching the fabric using the 2D pattern (Figure 8). The system displays how patches are connected by connectors or paired numbers [MI07]; information about connectors is useful to clarify the relationships shown on the screen, and numbers are useful as a printed reference on each patch for sewing. The system provides an automatic layout and manual arrangement interface for preparing the final pattern to be printed.

4. Shrink-free Flattening

This section describes the algorithm for flattening each cloth patch under the constraint that each edge of the flattened 2D mesh is always longer than that of the original 3D mesh. This constraint was based on our observation that a patch smaller than the target surface fails to cover the entire area, whereas a larger patch successfully covers the target area but with extra wrinkles.

4.1. Algorithm

The system first flattens the surface using a standard flattening algorithm and then iteratively refines the result so that it satisfies the shrink-free constraint. Initial flattening can apply any free boundary flattening method, such as LSCM [LSNJ02] or ABF++ [SLMB05]. Currently our system uses the as-rigid-as-possible flattening method [II09]. Basically, it applies the algorithm introduced in as-rigid-as-possible shape manipulation [IMH05] to surface flattening. It minimizes shape differences between each triangle of the original 3D mesh and the corresponding triangle on the flattened 2D mesh using least-squares methodology while preserving the mesh topology. The system first computes a mapping that allows free scaling and then adjusts the scale in the second step, which corresponds to an iteration step described in Liu et al. [LZX*08].

During the refinement, the system repeatedly applies the second step with varying edge weights to ensure that all edges of the flattened 2D mesh are equal to or longer than the corresponding edges of the 3D mesh. At each step, the system examines the 2D edge lengths in the previous iteration and updates the weight based on a function that gives more weight to shortened edges (Figure 9). This function gives a weight of 1 to elongated edges and 1000 to shortened edges, with a smooth transition between the two. It also updates each edge's estimated rotation, as discussed by Liu et al. [LZX*08]. The system terminates the iteration when all edges satisfy the constraints or all problematic edges have been weighted. Figure 10 shows the iterative refinement process.

A few details remain to be discussed. First, it is necessary to constrain at least two points to apply least-squares optimization. Currently our system constrains the edge at the graph center of the patch mesh. The two end points are fixed in the 2D space at a distance equal to that in the original 3D space. This is based on the observation that the center of the mesh is generally compressed and the periphery of the mesh is generally stretched. Our future research will focus on a way to avoid constraining specific vertices [MtAD08].

Second, assigning a high edge weight does not make the resulting 2D edge length exactly equal to the original 3D edge length; it only approaches the target length in terms of least squares. When an assigned weight is larger, the result is nearer the target. Small shrinkage is not a problem in practice, and the system judges the constraint to be satisfied when the amount of shrinkage is sufficiently small ($2D\_edge\_length > 3D\_edge\_length \times 0.94$ in our current prototype).

Third, there is no guarantee that the algorithm described above will satisfy all constraints; it is possible that an edge will need to be shortened even with high weights after many iterations. However, our experiments did not reveal any failures of this type; all edges satisfied the constraint within several iterations (Figure 11). The key to success appears to be the design of the weight function (Figure 9). We tested various types of functions with different parameter values, and our empirical results indicate that this particular function is effective.

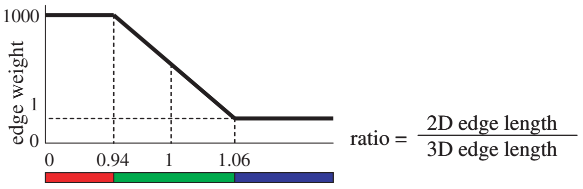


Figure 9: *Edge weight function.*

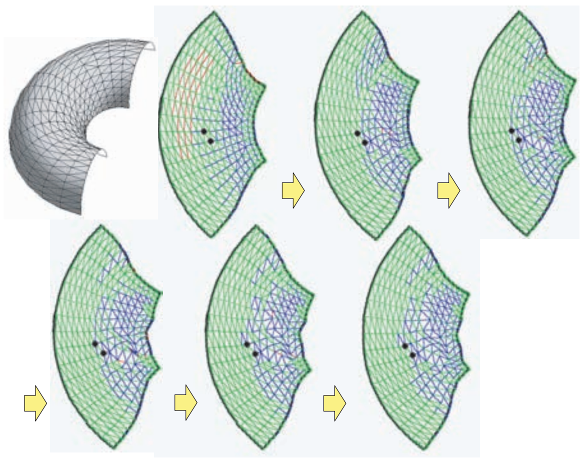
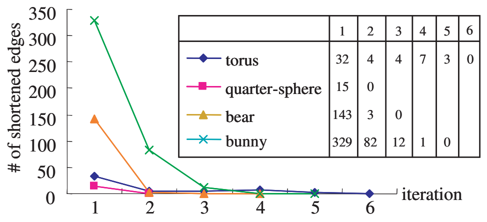


Figure 10: *Iteration process: shortened edges (red edges) gradually disappear; black dots indicate constrained vertices. See Figure 9 for color coding.*



	1	2	3	4	5	6
torus	32	4	4	7	3	0
quarter-sphere	15	0				
bear	143	3	0			
bunny	329	82	12	1	0	

Figure 11: *Number of shortened edges after each iteration.*

4.2. Results

Figure 12 compares results using angle-based flattening [SLMB05], one of the most popular flattening methods, and our shrink-free flattening. We first segmentation convex patches manually and apply our flattening method to the patches. The results show that our method successfully made all 2D edges be equal to or longer than the original 3D edges with small tolerance levels. In contrast, angle-based flattening caused about half of the edges to be shortened. It is possible to enlarge this result to enforce the constraint, but this would also cause excessive stretching of already long edges. Table 1 summarizes the distortion measures defined in Sander et al. [SSGH01]. The results of our method are comparable to a state-of-the-art method and still satisfied non-shrinking constraints. Our method is slower because it

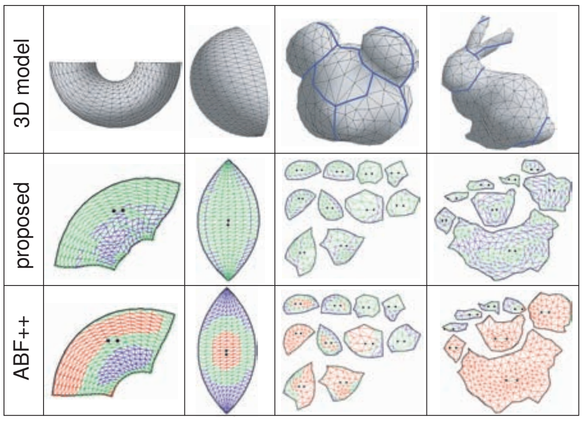
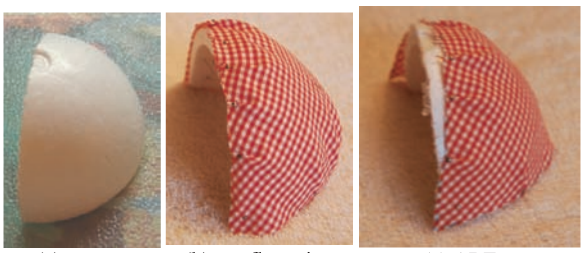


Figure 12: *Comparison of our flattening method and ABF++. See Figure 8 for color coding.*



(a) quarter sphere object (b) our flattening method (c) ABF++

Figure 13: *Comparison using real fabric: (b) our method; (c) ABF++ results.*

Table 1: *Comparison of distortion measures.*

	#vertex	Time (ms)	Edge	Angle	Area
proposed	256	1705	0.069	0.078	2.096
ABF++		177	0.083	0.018	2.091
proposed	281	931	0.032	0.038	2.992
ABF++		204	0.125	0.016	2.996
proposed	253	352	0.055	0.066	4.012
ABF++		92	0.073	0.059	4.099
proposed	303	1117	0.154	0.141	5.063
ABF++		108	1.995	0.104	5.308

solves a sparse linear system multiple times, but it still terminates within a few seconds for small meshes.

We also compared these two methods using real fabric (Figure 13). We created a 3D geometry corresponding to an object shaped like a quarter-sphere, flattened it, cut the fabric according to the results, and pasted it to the object surface. The cloth patch computed by our method successfully covered the surface using the correct amount of fabric. In contrast, the cloth patch computed by ABF++ was too short on the side and too long at the top and bottom. Simply scaling the result of ABF++ certainly covers the missing side area but even more surplus appears at the top and the bottom.

5. Uncovering

This section describes the algorithm for checking whether the object can be removed from the cover through the given opening. This is a difficult problem because it involves not only the opening but also the entire cover geometry. We can obtain an approximate answer by mimicking the uncovering process and examining its physical plausibility. The system virtually "pushes" vertices on the opening along the object's surface away from the opening while avoiding excessive stretching of cloth. The system continuously monitors the uncovering progress and terminates the process if all the cover material is on one side of the object's surface (success) or if it detects that the object is stuck (failure).

The uncovering test uses the resampled convex hull G of the target model M (see Figure 4b) as the object to be removed from the cover. This makes the uncovering process much easier and more stable than directly examining a complicated target model M with many bumps and concavities. Because the convex hull G always encloses the target model M, this uncovering test always gives a conservative answer. If the virtual uncovering is successful, the actual uncovering will also be successful.

5.1. Algorithm

This test uses an iterative process. In each iteration cycle, the system first expands the opening by moving its vertices in the direction perpendicular to the boundary curve along the convex hull surface G. The system then applies relaxation to the cover C to prevent excessive stretch; we applied the relaxation algorithm described in Igarashi and Hughes [IH02]. The system examines each cover triangle and moves the vertices of the triangle to recover its resting shape. The system also tracks the nearest point on the convex hull surface G to the cover vertex to prevent penetration. If the cover vertex is inside the convex hull G, the system pushes the cover vertex out of it.

Simply advancing the opening boundary and relying on relaxation cause local wrinkles and folds around the opening (Figure 14a). Therefore, the system explicitly moves all of the cover vertices together with the nearest vertex on the opening boundary (Figure 14b). Igarashi and Hughes used a similar technique to propagate the user's dragging operation to the entire cloth [IH02]. They adjusted the movement vectors so that they were always tangent to the object surface (Figure 14c). In contrast, we simply use an unconstrained 3D vector (Figure 14d) because our final goal is to remove the cover from the surface rather than slide the cover around on the object surface.

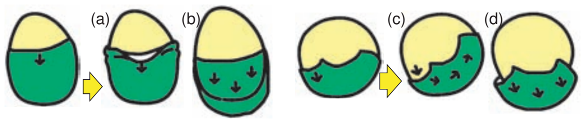


Figure 14: *(a) Moving a single vertex; (b) moving the entire vertices; (c) moving along the surface; (d) moving in one direction.*

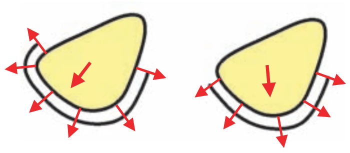


Figure 15: *Detecting the completion of uncovering: not completed (left); completed (right).*

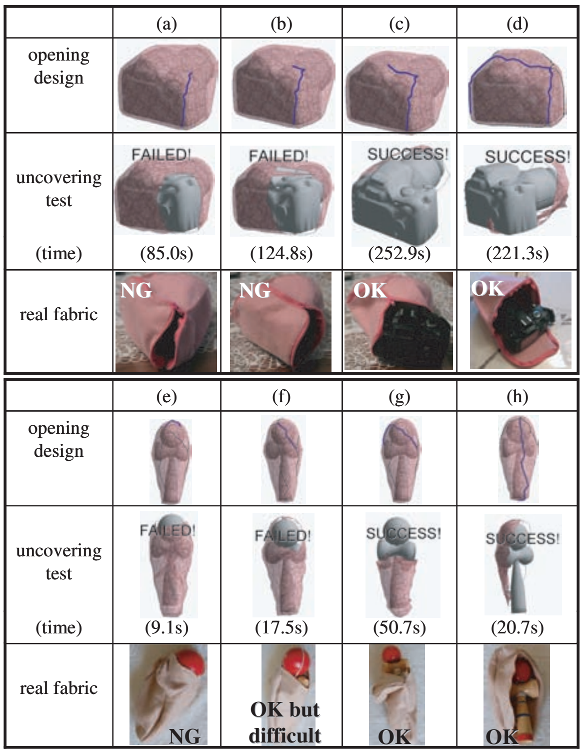
To detect the completion of the uncovering process, the system continuously monitors its progress by checking whether each vertex of the resampled convex hull surface G is still covered. The system emits a lay from the surface vertex in its normal direction and detects an intersection between the lay and the cover C. If the system detects an intersection, it identifies whether the vertex is still covered. The uncovering terminates when all vertices of the covered surface vertices are on one side of the convex hull G (Figure 15). Specifically, the system computes the average of the covered vertices' normal and the angle difference between the average and each normal. The system judges that convex hull G has been successfully uncovered when the differences are all smaller than 90 degrees. This simple method works well because the system uses a convex hull.

The detection of a stuck is more complex. Currently the system monitors the uncovering progress and identifies the object as being stuck when the number of covered vertices does not decrease over a specified number of iterations (10 in this prototype).

5.2. Results

The algorithm involves various simplifications and very approximate in nature. It assumes relatively simple opening and can behave unexpectedly when a complicated opening is given. For example, when the opening forms an almost-closed loop (e.g. "C" shape), the behavior of the cover surface inside of the loop becomes unstable because the vertices gather at the center (Figure 16d). The system still produces a reasonable result because the problematic area gradually shrinks and is considered as uncovered. We plan to explicitly identify an almost-closed loop in an opening and treat it as a loop in the future.

As we have already discussed, our algorithm is conservative in nature; success in the virtual uncovering test means that uncovering will be successful in the real world, because the test applies the convex hull G instead of the model's actual geometry M. On the other hand, even if the virtual uncovering is unsuccessful, uncovering may be successful in the real world.



	(a)	(b)	(c)	(d)
opening design				
uncovering test (time)	FAILED! (85.0s)	FAILED! (124.8s)	SUCCESS! (252.9s)	SUCCESS! (221.3s)
real fabric	NG	NG	OK	OK

	(e)	(f)	(g)	(h)
opening design				
uncovering test (time)	FAILED! (9.1s)	FAILED! (17.5s)	SUCCESS! (50.7s)	SUCCESS! (20.7s)
real fabric	NG	OK but difficult	OK	OK

Figure 16: *Virtual and actual uncovering tests. We used 1.5 GHz Intel Core2 PC (RAM 1.5MB).*

Figure 16 shows the results of virtual and actual uncovering. Despite the limitations discussed above, our algorithm generally yielded correct results. We did observe a slight mismatch between predicted and actual results, as shown in Figure 16f. However, pulling this object from its cover in the real world required a lot of work-pushing and pulling the object inside the cover. Therefore, the results generally indicate that the current uncovering test is sufficiently accurate for use as a guiding tool for non-professional users.

6. User Experiences

We provided the system to two test users and asked them to design their own covers. The user 1 is not familiar with computers but good at sewing. In contrast, the user 2 has a lot of experience with computer graphics but has little sewing experience. Figure 17 shows some of the results. The test users understood how to use the system within 5 to 10 min and had generally completed their designs within 5 to 20 min. It took them 1 to 2 hours to sew a cover. They generally

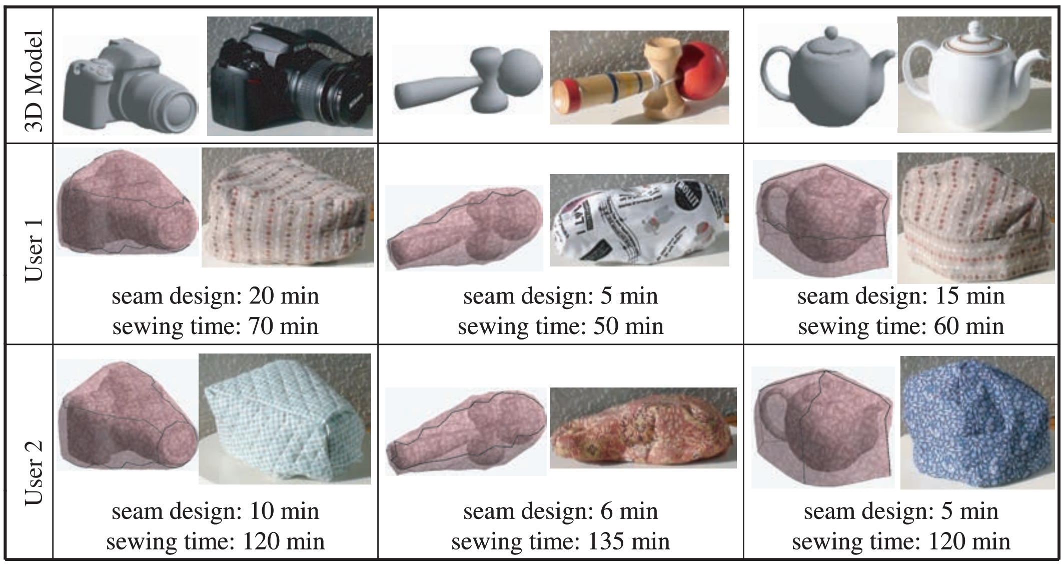


Figure 17: *Real covers designed by the test users using our system.*

found the system easy to use and enjoyed the experience; they particularly appreciated the ability to check whether the opening was large enough to remove the object before sewing a real fabric cover. They also appreciated the default automatic cover segmentation and said that it was a good way to demonstrate how to design an original segmentation. They provided feedback about improvements, including the inclusion of auxiliary functions such as the ability to design symmetric parts, e.g., having a segmentation designed on one side of the cover to appear on the other side. They also expressed a desire to design an original cover geometry, possibly using a simple sketching program such as Plushie [MI07].

7. Limitations and Future Work

The technical problems discussed in this paper are not clearly defined, and the validation of the proposed methods was rather qualitative. We believe that our findings are acceptable as an initial experiment, because our goal is to provide a practical solution to a complicated real-world problem rather than provide an analytic solution to a theoretical problem. Still, it should be possible to conduct a more rigorous analysis of some aspects of the system. For example, our shrink-free flattening algorithm is not guaranteed to converge and it is only quasi shrink-free. We only empirically validated its convergence in most examples, but more theoretical analysis is desired. Similarly, the uncovering test is very limited because we only focus on the vertices and do not check face-face collision. This actually can cause fail in some cases and better solutions need to be developed in the future.

The uncovering test is too slow. The current implementation is not yet optimized for speed and we can easily obtain reasonable speed-up by tweaking details such as to adjust parameters. However, we ideally want to provide interactive feedback, that is, to continuously run the test in the background and show the result while the user is painting the opening. We need to have a significantly different strategy to achieve this such as to apply a sophisticated structure analysis to the model and the cover before running the test.

Our current system is designed for rigid objects. Future work will focus on the design of covers for soft objects such as stuffed animals and articulated objects with joints, such as robot toys. We also plan to work on tight fitting, non-convex covers for objects such as globes and violins. Other flattening methods such as authalic parameterization [DMA02] might be worth testing. It would also be interesting to extend our system to design covers (clothing) for living things such as babies and pets.

This research shows that incorporating physical constraints into a computer model used for interactive editing can be very helpful for non-professional users who are designing functioning real-world objects. We plan to apply this approach to various other problems, e.g., analyzing liquid flow to help design a better teapot, or predicting possible wear to leather to help design a sturdy briefcase.

Acknowledgments

We would like to thank Dr. Yutaka Ohtake, The University of Tokyo for his helpful comments and advice. We also thank the test users. This work was partially supported by Grant-in-Aid for JSPS Fellows (20-10073).

References

[3ds] Autodesk. 3ds Max. http://usa.autodesk.com/. 2

[AFS06] ATTENE M., FALCIDIENO B., SPAGNUOLO M.: Hierarchical mesh segmentation based on fitting primitives. *The Visual Computer 22*, 3 (2006), 181–193. 4

[BFA02] BRIDSON R., FEDKIW R., ANDERSON J.: Robust treatment of collisions, contact and friction for cloth animation. In *ACM SIGGRAPH 2002* (2002), pp. 594–603. 2

[CSAD04] COHEN-STEINER D., ALLIEZ P., DESBRUN M.: Variational shape approximation. *ACM Transactions on Graphics 23*, 3 (2004), 905–914. 3, 4

[das] Dassault Systemes. DELMIA. http://www.delmia.com/. 2

[DJW*06] DECAUDIN P., JULIUS D., WITHER J., BOISSIEUX L., SHEFFER A., CANI M. P.: Virtual garments: A fully geometric approach for clothing design. *Computer Graphics Forum (Proceedings of Eurographics 2006) 25*, 3 (2006), 625–634. 2

[DMA02] DESBRUN M., MEYER M., ALLIEZ P.: Intrinsic parameterizations of surface meshes. *Computer Graphics Forum 21*, 3 (2002), 209–218. 8

[efp] EfPiSoft. http://efpisoft.sourceforge.net/. 4

[EM94] EDELSBRUNNER H., MUECKE E. P.: Three-dimensional alpha shapes. *ACM Transactions on Graphics 13*, 1 (1994), 43–72. 3

[GWH01] GARLAND M., WILLMOTT A., HECKBERT P. S.: Hierarchical face clustering on polygonal surfaces. In *ACM Symposium on Interactive 3D Graphics* (2001), pp. 49–58. 4

[IH02] IGARASHI T., HUGHES J. F.: Clothing manipulation. In *Proceedings of 15th Annual Symposium on User Interface Software and Technology (Proceedings of ACM UIST 2002)* (2002), pp. 91–100. 2, 4, 6

[II09] IGARASHI T., IGARASHI Y.: Implementing as-rigid-as-possible shape manipulation and surface flattening. *Journal of Graphics Tools* (2009). 5

[IIS08a] IGARASHI Y., IGARASHI T., SUZUKI H.: Knitting a 3d model. *Computer Graphics Forum (Proceedings of Pacific Graphics 2008) 27*, 7 (2008), 1737–1743. 2

[IIS08b] IGARASHI Y., IGARASHI T., SUZUKI H.: Knitty: 3D modeling of knitted animals with a production assistant interface. In *Eurographics 2008 Annex to the Conference Proceedings* (2008), pp. 187–190. 2

[IMH05] IGARASHI T., MOSCOVICH T., HUGHES J. F.: As-rigid-as-possible shape manipulation. *ACM Transactions on Computer Graphics (Proceedings of SIGGRAPH 2005) 24*, 3 (2005), 1134–1141. 5

[JKS05] JULIUS D., KRAEVOY V., SHEFFER A.: D-Charts: quasi developable mesh segmentation. *Computer Graphics Forum (Proceedings of Eurographics 2005) 24*, 3 (2005), 981–990. 2, 3, 4

[KL00] KUFFNER J. J., LAVALLE S. M.: RRT-Connect: An efficient approach to single-query path planning. In *Proceedings of IEEE International Conference on Robotics and Automation (ICRA'2000)* (2000), pp. 995–1001. 2

[LA06] LIEN J.-M., AMATO N.: Approximate convex decomposition of polyhedra. In *Technical Report TR06-2002, Texas A&M University* (2006). 3

[LCWK07] LU L., CHOI Y.-K., WANG W., KIM M.-S.: Variational 3D shape segmentation for bounding volume computation. *Computer Graphics Forum 26*, 3 (2007), 329–338. 3

[Llo57] LLOYD S. P.: Least squares quantization in PCM. *Bell Telephone Laboratory Memorandum. reprint IEEE Transactions on Information Theory IT-28*, 2 (1957), 129–I37. 3, 4

[LSNJ02] LÉVY B., SYLVAIN P., NICOLAS R., JEROME M.: Least squares conformal maps for automatic texture atlas generation. In *Proceedings of ACM SIGGRAPH 2002* (2002), pp. 362–371. 2, 4

[LZX*08] LIU L., ZHANG L., XU Y., GOTSMAN C., GORTLER S. J.: A local/global approach to mesh parameterization. *Computer Graphics Forum (Eurographics Symposium on Geometry Processing 2008) 27*, 5 (2008). 2, 5

[may] Autodesk. Maya. http://usa.autodesk.com/. 2

[MHS99] MCCARTNEY J., HINDS B. K., SEOW B. L.: The flattening of triangulated surfaces incorporating darts and gussets. *Computer-Aided Design 31*, 4 (1999), 249–260. 2

[MI07] MORI Y., IGARASHI T.: Plushie: An interactive pattern design for plush toys. *ACM Transactions on Graphics (Proceedings of SIGGRAPH 2007) 23*, 3 (2007), Article No.45. 2, 4, 8

[MS04] MITANI J., SUZUKI H.: Making papercraft toys from meshes using strip-based approximate unfolding. *ACM Transactions on Graphics (Proceedings of SIGGRAPH 2004) 23*, 3 (2004), 259–263. 2

[MtAD08] MULLEN P., TONG Y., ALLIEZ P., DESBRUN M.: Spectral conformal parameterization. In *Eurographics Symposium on Geometry Processing 200* (2008). 5

[rob] Siemens PLM Software. Tecnomatix ROBCAD. http://www.plm.automation.siemens.com/. 2

[SLMB05] SHEFFER A., LÉVY B., MOGILNITSKY M., BOGOMYAKOV A.: ABF++: Fast and robust angle based flattening. *ACM Transactions on Graphics 24*, 2 (2005), 311–330. 2, 4, 5

[SSGH01] SANDER P. V., SNYDER J., GORTLER S. J., HOPPE H.: Texture mapping progressive meshes. In *ACM SIGGRAPH 2001 Conference Proceedings* (2001), pp. 409–416. 5

[Tur92] TURK G.: Re-tiling polygonal surfaces. *SIGGRAPH Comput. Graph. 26*, 2 (1992), 55–64. 3

[VCT95] VOLINO P., COURCHESNE M., THALMANN N. M.: Versatile and efficient techniques for simulating cloth and other deformable objects. In *ACM SIGGRAPH 95* (1995), pp. 137–144. 2

[Wan08] WANG C. C. L.: WireWarping: a fast surface flattening approach with length-preserved feature curves. *Computer-Aided Design 40*, 3 (2008), 381–395. 2

[YBS08] YOSHIZAWA S., BELYAEV A., SEIDEL H.-P.: A fast and simple stretch-minimizing mesh parameterization. In *Proceedings of Shape Modeling and Applications* (2008), pp. 200–208. 2

[ZHKM08] ZHANG L., HUANG X., KIM Y. J., MANOCHA D.: D-Plan: Efficient collision-free path computation for part removal and disassembly. *Computer-Aided Design and Applications 6*, 6 (2008), 774–786. 2

[ZKVM06] ZHANG L., KIM Y. J., VARADHAN G., MANOCHA D.: Generalized penetration depth computation. In *ACM Solid and Physical Modeling Conference (SPM06)* (2006), pp. 173–184. 2

[ZX06] ZHONG Y., XU B.: A physically based method for triangulated surface flattening. *Computer-Aided Design 38* (2006), 1062–1073. 2