Plushie: An Interactive Design System for Plush Toys

Yuki Mori*
The University of Tokyo

Takeo Igarashi*
The University of Tokyo / JST, PRESTO

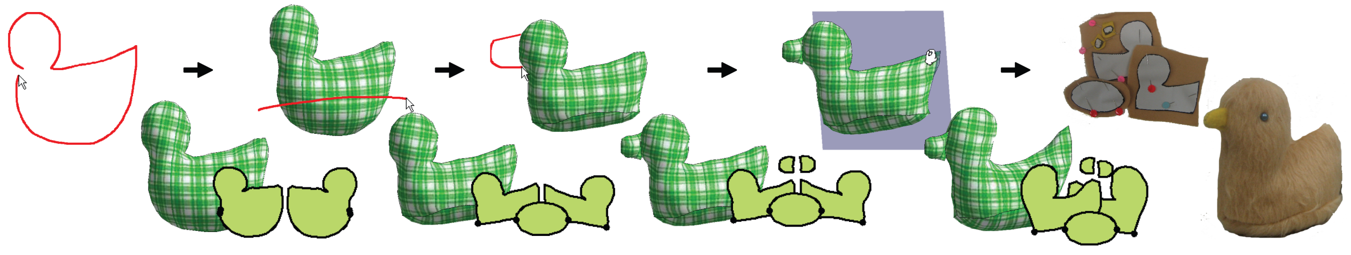


Figure 1: *Designing an original plush toy using our system. The user interactively edits the 3D model on the screen using a sketching interface. Internally, the system generates 2D cloth pattern and shows the 3D model as a result of applying simple simulation to the pattern.*

Abstract

We introduce Plushie, an interactive system that allows nonprofessional users to design their own original plush toys. To design a plush toy, one needs to construct an appropriate two-dimensional (2D) pattern. However, it is difficult for non-professional users to appropriately design a 2D pattern. Some recent systems automatically generate a 2D pattern for a given three-dimensional (3D) model, but constructing a 3D model is itself a challenge. Furthermore, an arbitrary 3D model cannot necessarily be realized as a real plush toy, and the final sewn result can be very different from the original 3D model. We avoid this mismatch by constructing appropriate 2D patterns and applying simple physical simulation to it on the fly during 3D modeling. In this way, the model on the screen is always a good approximation of the final sewn result, which makes the design process much more efficient. We use a sketching interface for 3D modeling and also provide various editing operations tailored for plush toy design. Internally, the system constructs a 2D cloth pattern in such a way that the simulation result matches the user's input stroke. Our goal is to show that relatively simple algorithms can provide fast, satisfactory results to the user whereas the pursuit of optimal layout and simulation accuracy lies outside this paper's scope. We successfully demonstrated that non-professional users could design plush toys or balloon easily using Plushie.

CR Categories: I.3.6 [Computer Graphics]: Methodology and Techniques—Interaction Techniques; I.3.5 [Computer Graphics]: Computational Geometry and Object Modeling—Geometric Algorithms

Keywords: plush toys, sketch-based modeling, cloth simulation

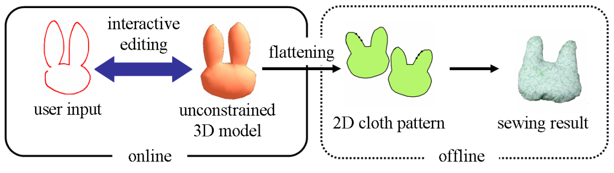
*e-mail: {yukim, takeo}@acm.org

1 Introduction

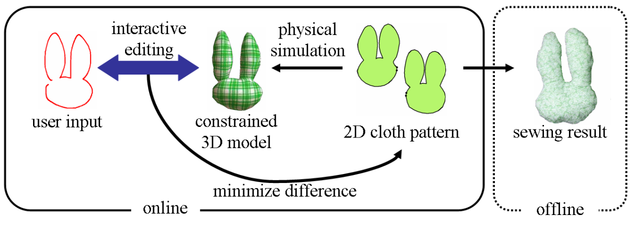
A computer can be a powerful tool for designing real-world artifacts. One can build a virtual model in a computer and use it to run various simulations, without the need to build and damage costly real objects. The benefits are evident in many areas, from architecture to automobile design. In these traditional applications, modeling and simulation have been completely separated. A virtual model is created in three-dimensional (3D) modeling software without considering any physical constraints, and then passed later to a simulation environment. If the simulation result reveals a problem, the user returns to modeling to fix the problem. This can be more efficient if the system runs the simulation concurrently with modeling, so that only models that are physically realizable are created. In this way, the user can more efficiently explore the design dimensions within realistic constraints. From the user's point of view, the model generated by the system may not exactly take his/her input shape but takes a physically realizable shape reflecting the input shape.

Some recent systems have tried to incorporate physical simulation into an interactive design process. Igarashi and Hughes [2002] developed a mark-based interface for putting clothing on a virtual character, and Decaudin et al. [2006] proposed a system for designing an original garment via sketching. Both used simple geometric simulations to represent the physical properties of cloth material. Masry and Lipson [2005] described a system in which the user can quickly build a CAD model via sketching and immediately apply finite element analysis to the model. However, model construction is computed before simulation in these systems, and no dynamic feedback loop exists between the simulation result and the original user input.

We are experimenting with modeling guided by a concurrent simulation in plush toy design. Plush toys are one of the most familiar objects in our daily life but their design is difficult. One must design an appropriate two-dimensional (2D) pattern to obtain a particular 3D shape, but the relationship between the two is nontrivial, and intensive experience and knowledge is required to do so appropriately. As a result, most people simply buy ready-made toys and do not enjoy the design and construction of their own plush toys. We have tried to enable these people to design their own toys by providing an easy but powerful modeling tool that combines a sketching interface and integrated physical simulation.



(a) Traditional framework



(b) Our framework

Figure 2: *Designing a plush toy with a computer. Traditional approach (a) first constructs an unconstrained virtual model and generates a 2D pattern by applying segmentation and flattening. The final sewn result can be very different from the virtual model. Our system (b) directly generates 2D cloth pattern such that the simulation result matches the user input. In this way, the virtual model is always a good estimation of the final sewn result.*

Methods for making plush toys from a given 3D model have been proposed in recent years. Mitani and Suzuki [2004] and Shatz et al. [2006] presented automatic segmentation of a 3D model into developable patches for constructing paper craft models. Similarly, Julius et al. [2005] proposed automatic segmentation and flattening of a model for plush toys. One problem of an automatic approach that relies on purely geometric criteria is that it is difficult to perceptually capture important features such as symmetry. Mori and Igarashi's [2006] system helps manual segmentation of a model by providing automatic flattening and showing the result of physical simulation. These systems make plush toy design more accessible, but the fundamental challenge of creating an original plush toy is still unresolved. One can generate simple models using some modeling software (e.g., [Igarashi et al. 1999; Karpenko and Hughes 2006]) but an arbitrary 3D model is not necessarily realizable as a real plush toy, and the final sewn result can be very different from the original 3D model (Figure 2 a).

Our system, Plushie, allows the user to create a 3D plush toy model from scratch by simply drawing its desired silhouette. The user can also edit the model, such as by cutting the model and adding a part, using simple sketching interface. The resulting model is always associated with a 2D pattern and the 3D model is the result of a physical simulation that mimics the inflation effect caused by stuffing. Therefore, the model on the screen is always a good estimation of the final sewn result (Figure 2 b). Internally, the system computes the geometry of the 2D pattern so that the simulation result matches the user's input sketch. This is a nontrivial inverse problem and we address this by using simple iterative adjustment method. We show that a very simple simulation method, just moving vertices to their normal directions to mimic pressure and pulling them back to maintain edge length, works well for our application and provides an unprecedented experience of designing a physical object in a computer. We run a workshop in a museum to have novice users try our system and have observed that even children can design their own plush toys.

Our contribution is in the overall design of the interactive system. We use relatively simple algorithms as a proof of concept to provide immediate feedback to the user. More sophisticated, off-line algorithms for texture atlas generation [Milenkovic 1999; Bruno et al. 2002] and cloth simulation [Grinspun et al. 2002; Choi and Ko 2002; Breen et al. 1994] have been studied in textile industry. It is our future work to explore a way to apply these sophisticated methods to interactive setting.

2 User Interface

The system consists of two windows: one shows the 3D plush toy model being constructed and the other shows the corresponding 2D pattern (3). The user works on the 3D view, interactively building the 3D model by using a sketching interface. The 2D view is mainly for reference but the user can also edit the 2D pattern directly when desired. The 3D model is produced from a physical simulation of the assembled 2D pattern. After each input from the user, the system updates the 2D pattern so that the simulation result matches the user input. This guarantees that the model is always realizable as a real plush toy and that the 2D pattern is readily usable as a template for cutting and sewing real fabric.

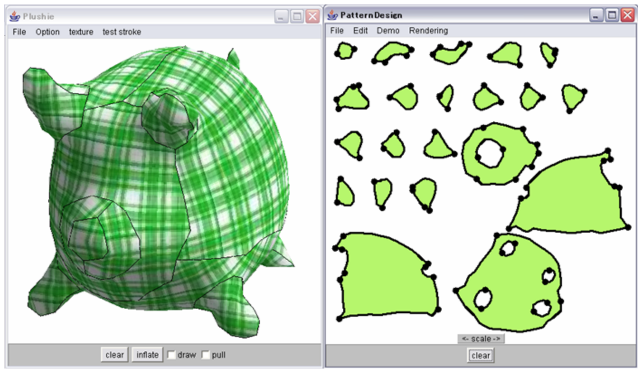


Figure 3: *A screen snapshot of the Plushie system.*

2.1 3D Modeling Operations

The modeling operations are based on gestural interface introduced by Igarashi et al. [1999]. The user interactively draws free-form strokes on the canvas as gestures and the system performs corresponding operations. We also provide some special editing operations tailored for plush toy design.

Creating a New Model: Starting with a blank canvas, the user creates a new plush toy model by drawing its silhouette as a closed free-form stroke. The system automatically generates two cloth patches corresponding to the stroke and visualizes the shape of the resulting plush toy by applying a simple physical simulation (Figure 1 a).

Cut: A cut operation makes relatively flat surfaces, such as those in a foot or belly. A cutting stroke should start outside of the model, cross it, and end outside of the model (Figure 1 b). The model is cut at the intersection and flat patch is generated at the cross-section.

Creation of a Part: The user can add protruding parts such as the ears and arms to the base model by drawing a single stroke that defines the silhouette of the part. The stroke should start and end on the base model (Figure 4 a). The system generates two candidate shapes and presents them to the user as suggestions [Igarashi and Hughes 2001](Figure 4 b). One is for fat, rounded parts like the body, arm, and leg (Figure 4 c). Their base is connected to the

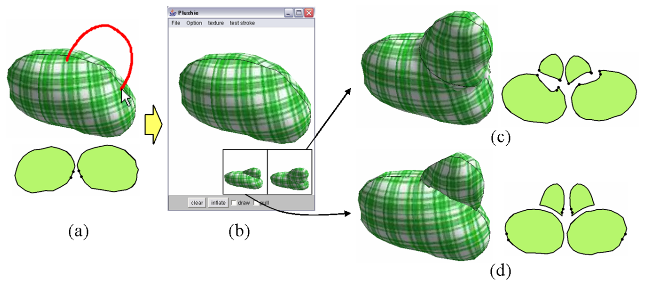


Figure 4: *User interface of part creation. (a) The user draws a stroke and (b) the system suggests two different possibilities. The user chooses one (c, d).*

base model with an open hole. The other candidate shape is for thin parts like ears and the tail, whose base is closed (Figure 4 d). The user clicks the desired thumbnail and the system updates the main model accordingly. We found that the ability to create thin parts with a single stroke is particularly useful. They are frequently seen in real toys and are difficult to design using standard modeling software. Figure 18 shows a couple of example models with thin parts.

Pull: The user can grab a seam line and pull it to modify the shape. For example, the user can pull an ear to make it larger when it is smaller than the other (Figure 5). The pulling operation begins when the user starts dragging on the background region near a seam line. The system changes the mouse cursor when it approaches a seam line to indicate that the user can start pulling. We use the peeling interface introduced by Igarashi et al. [2005] to adjust the size of the region to be deformed; that is, the more the user pulls, the larger the area of the deformed region. The system continuously updates the 2D cloth pattern during pulling and shows the simulation result in the 3D view.

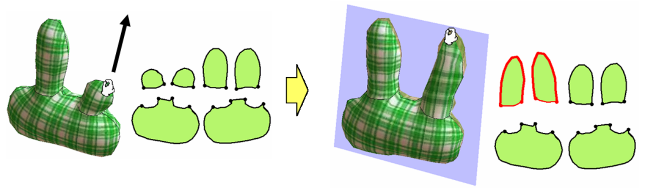


Figure 5: *User interface of the pull operation.*

Insertion and Deletion of Seam Lines: The modeling operations performed thus far automatically generate 2D patches according to predefined algorithms and seam lines (patch boundaries) appear on the 3D model surface without the user's explicit control. However, it is sometimes desirable for knowledgeable users to design seam lines manually, for more detailed control. This is especially important when using non-stretchy cloth as in balloon models because one needs to divide a rounded surface into many almost-developable small patches (Figure 17 bottom).

The user can add a new seam in the seam line drawing mode by drawing a free-form stroke on the model surface (Figure 6). The corresponding cloth patch is then automatically cut along the new seam line. If the stroke crosses the entire patch, the patch is divided into two separate patches. If the stroke starts or ends in the middle of a patch, it becomes a dart. The 3D geometry does not change immediately after the insertion of these seam lines, but the user can pull the seam line afterwards to modify the shape. This operation is very useful for creating a salient feature in the middle of a flat

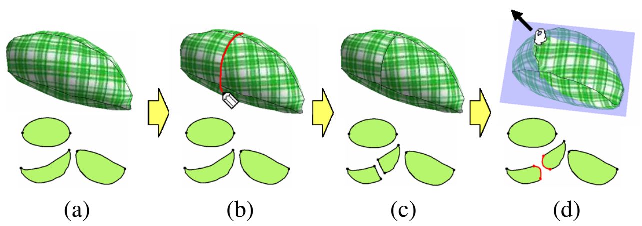


Figure 6: *Insertion of a seam line. (a) Before drawing a line. (b) After drawing a line. (c) The seam line's two endpoints snap at other seam lines. (d) After pulling the seam line.*

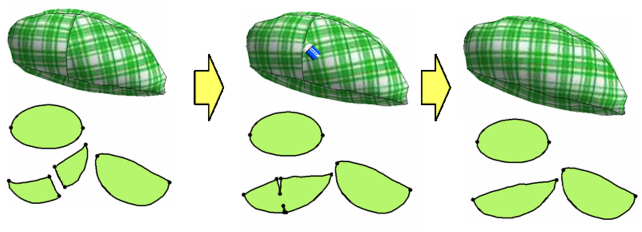


Figure 7: *Deletion of a seam line.*

patch. Deletion is achieved by tracing the target seam line in the erasing mode. This merges the separated patches together and thus flattens the area (Figure 7).

2.2 Operations on the 2D Pattern View

The 2D pattern view is mainly used to preview the pattern to be printed for sewing, but it also works as an interface for advanced users to edit the pattern directly. The preview helps the user to understand the relationship between the 3D model and 2D patches and gives a sense of the labor required for assembling the patches. The system can display how patches are connected by showing connectors or paired numbers (Figure 8). Connectors are useful for understanding the relationship on the screen and numbers are useful as a printed reference on each patch. The system provides an automatic layout and manual arrangement interface for preparing the final pattern to be printed.

The system also allows the user to edit the patches directly by using the pulling interface. The user can grab the boundary of a patch and pull it to deform the shape [Igarashi et al. 2005]. We again use a peeling interface to adjust the size of area to be deformed. The effect of 2D deformation immediately appears in the 3D view because of the physical simulation. The ability to deform an individual patch is useful for designing asymmetric shapes such as a penguin body (Figure 9, Figure 18). The pull operation is also useful for opening a dart line to make a flat patch swell more (Figure 10).

3 Implementation

We use standard triangle mesh for the representation of 3D model and the 2D patches. We use relatively coarse mesh (1000-2000 vertices) to achieve interactive performance. Each vertex, edge, and face of the 3D mesh is associated with corresponding entities in the 2D mesh. A 3D mesh is always given as a result of applying a physical simulation to the assembled 2D pattern. To be more precise, the physical simulation applied to the 3D mesh is governed by the rest length of each edge, which is defined in the 2D mesh geometry. For each modeling operation, the system constructs the

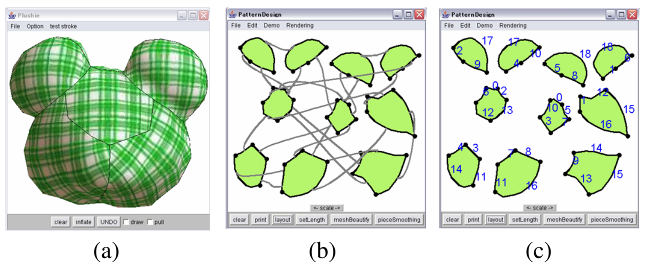


Figure 8: *Patches connected to each other using connectors (b) and numbers (c).*

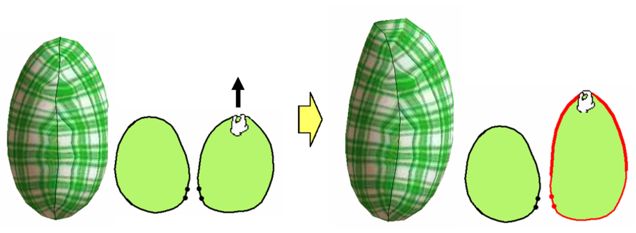


Figure 9: *Pulling a 2D patch.*

initial 2D patches and the 3D geometry corresponding to the input stroke, and then runs a physical simulation to update the 3D geometry. The system then adjusts the patch shape so that the simulation results match the input strokes. This section describes these implementation details.

3.1 Physical Simulation

We use a simple static method for the physical simulation. We examined other, more elaborate methods, such as finite element methods [Grinspun et al. 2002], dynamic simulation [Choi and Ko 2002], and energy minimization [Breen et al. 1994], but we found that the simple approach is best suited for our purpose. It is easy to implement, fast enough for interactive modeling, and sufficiently robust for dealing with adverse user operations. More importantly, it produces a reasonable estimation of the resulting plush toy shape. As shown in Figure 14, it successfully reproduces characteristic behaviors seen in the stuffed cloth.

In each simulation cycle, the system first moves each face slightly in its normal direction to mimic the effect of internal pressure (Figure 11 a). The displacement of a vertex v_i is computed as a weighted sum of the neighboring faces (F_i)' displacements:

$$v_i \mathrel{+}= \alpha \frac{\sum_{f \in F_i} A(f)\, \mathbf{n}(f)}{\sum_{f \in F_i} A(f)} \tag{1}$$

where $A(f)$ is the area of a face f and $\mathbf{n}(f)$ is the normal of the face.

The system then adjusts the length of each edge to preserve the integrity of the cloth material [Desbrun et al. 1999] (Figure 11 b). We decided to prevent stretching only and tolerate compression because plush toys' rotund shape is generated from compression (small winkles) along the seam lines. The displacement of a vertex v_i is computed as a weighted sum of the forces (t_{ij}) from the neighboring edges (E_i):

$$v_i \mathrel{+}= \beta \frac{\sum_{e_{ij} \in E_i} \{A(e.\mathit{leftface}) + A(e.\mathit{rightface})\}\, t_{ij}}{\sum_{e_{ij} \in E_i} \{A(e.\mathit{leftface}) + A(e.\mathit{rightface})\}} \tag{2}$$

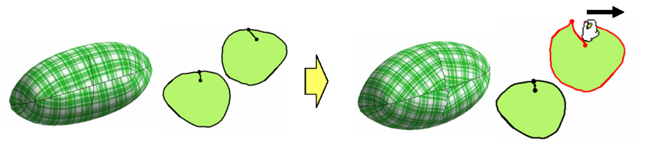


Figure 10: *Opening a dart line.*

$$t_{ij} = \begin{cases} 0.5 \cdot (v_j - v_i) \cdot \frac{|v_i - v_j| - l_{ij}}{|v_i - v_j|} & \text{if} \quad |v_i - v_j| \geq l_{ij} \\ 0 & \text{if} \quad |v_i - v_j| < l_{ij} \end{cases} \tag{3}$$

where l_{ij} represents the rest length of an edge e_{ij}.

The second part (adjustment of edge length) runs ten times in each cycle to prevent excessive stretch. It takes approximately 30 simulation cycles (2 seconds) to converge in our typical examples. The parameter setting in our current implementation is $a = 0.02$ and $b = 1$. Although it is possible to show the result only after the convergence, we decided to show the intermediate shape because test users preferred to see the inflation process.

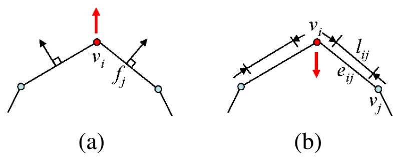


Figure 11: *Our simple model to mimic stuffing effect. (a) the system first moves each face to its normal direction slightly to mimic the effect of internal pressure. (b) the system adjusts the length of each edge to preserve the integrity of cloth material.*

3.2 3D Modeling

Creating a New Model: The input stroke is projected onto an invisible plane at the center of the world facing the screen, and the system generates an initial two-sided mesh inside of the closed region. Each side of the mesh is used directly as a 2D patch for the model. The system then applies the physical simulation to the mesh. It inflates the mesh to the direction perpendicular to the viewing direction, but its silhouette actually becomes smaller as it inflates (Figure 12). The system waits until the simulation converges and then starts to adjust the 2D pattern so that the simulation result matches the input stroke. Specifically, the system calculates the distance d_i from a vertex v_i of the 3D mesh along the seam line to the corresponding point p_i in the projected input stroke along its normal direction, and moves the corresponding 2D vertex u_i on the patch boundary in its normal direction by that amount d_i (Figure 12). We prevent self-intersection by detecting collision during vertex relocation. After modifying the patch boundary, the system updates the 2D mesh by applying Laplacian smoothing constraining the boundary vertices (Figure 13 d):

$$\arg\min_{v_i} \{ \textstyle\sum_{v_i} |v_i - \frac{1}{|N_i|} \sum_{v_j \in N_i} v_j|^2 + \sum_{v_i \in B} |v_i - v_i{}'|^2 \} \tag{4}$$

where N_i is the one ring neighbor of v_i and B is the boundary.

The length of the edges in the updated 2D mesh is then used as the new rest length in the simulation. The system repeats this adjustment process and the physical simulation until convergence. We also apply simple gaussian smoothing to the boundary curve once in

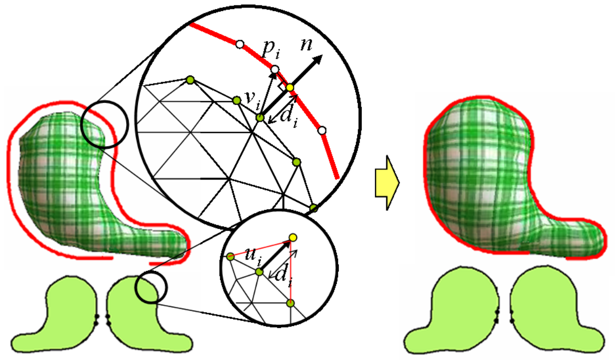


Figure 12: *Adjustment process after creation. The system enlarges the 2D pattern so that the simulation result matches the input stroke. The 2D boundary vertex (v) moves in its normal direction by the amount proportional to the distance between the corresponding 3D vertex and the input stroke.*

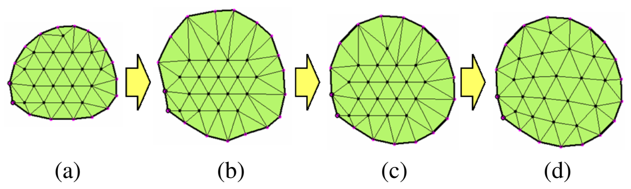


Figure 13: *Updating the 2D mesh geometry. (a) before deformation. (b) boundary vertices are moved. (c) smoothing is applied to the boundary. (d) Laplacian smoothing is applied to the internal vertices.*

every five iteration cycles to maintain smoothness along the boundary (Figure 13 c). It takes approximately 20 iterations (2 seconds) to converge in our typical examples.

This simple algorithm works well in practice for our application. Figure 14 shows some examples in which our algorithm successfully found appropriate 2D patches that yielded the desired 3D shapes. In some situations, the input shape is not realizable as a plush toy model consisting of two patches. For example, a sharp concavity is not realizable without causing self-intersection in the 2D patch. In these cases, the system terminates the optimization process, leaving a gap between the input stroke and the 3D model. This indicates that the desired shape is not possible with two patches. The user must add additional seam lines to obtain more control.

Cut: The system constructs a curved surface by sweeping the cutting stroke on the screen in the viewing direction and dividing the mesh along the surface. The right-hand side of the surface is removed and a new mesh is created on the cross-section. The cross-section is always developable, so the system simply flattens it and uses it as a 2D patch.

Creation of a Part: The system first projects the two endpoints of the input stroke onto the base model surface. A plane that passes through these 3D points and is facing toward the screen is constructed and the input stroke is projected onto it. The system then draws an ellipse on the model surface for constructing a fat part and draws a line for creating a thin part (Figure 15). The ellipse or the line (what we call base curves) is also projected to the plane. The system generates a 2D mesh on the projection plane in the area enclosed by the projected input stroke and the projected base curve. The 2D mesh is duplicated and serves as 2D pattern and as the

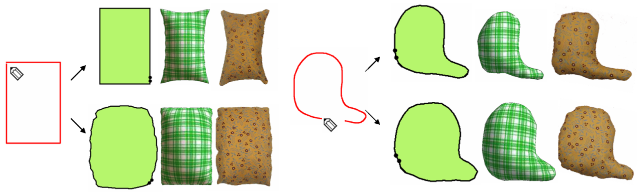


Figure 14: *Physical simulation and shape adjustment. The red lines indicate the input strokes. The top row shows the result of converting the input into patterns directly, and the bottom row shows the outcome when the adjustment process is applied to the patterns. The green shapes in the middle show the simulation results and the brown ones on the right show the real fabric models, both resulting from the 2D pattern on the left.*

initial 3D geometry for the added part. As in the initial model creation case, the flat two-sided 3D mesh is inflated by physical simulation. The silhouette of the added part gradually shrinks and the system enlarges the 2D pattern so that the silhouette matches the input stroke as in initial creation.

In case of a part with an elliptic base curve, the system cuts open the base surface and stitches it with the newly created mesh. The result is a single connected mesh, and physical simulation is applied uniformly to the entire mesh. On the other hand, the system does not open the base mesh in case of the linear base curve. The new part is created as an independent closed mesh and the simulation is applied separately to the base mesh and the new part. The base mesh inflates independently of the part mesh, and the base curve is treated as a positional constraint in the simulation of the part mesh (we simply do not move these vertices in the simulation cycle).

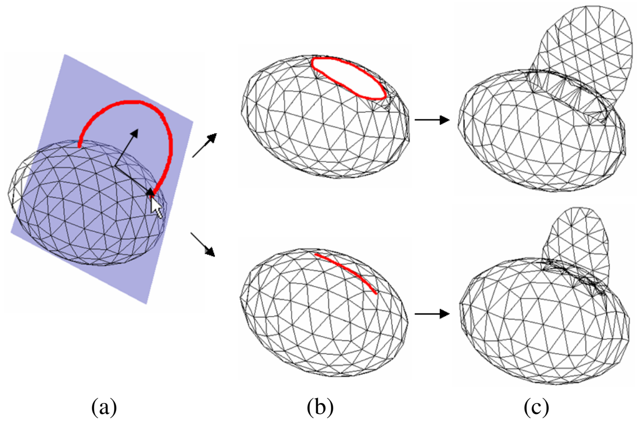


Figure 15: *Creation of a part. The system projects the input stroke to a working plane and cuts the base mesh with either an elliptic curve or a line (b). The 3D geometry is constructed by creating a mesh between the projected stroke and the base curves (c).*

Pull: The pull operation is a bit involved because the system cannot directly modify the 3D mesh and must do so indirectly by deforming the corresponding 2D pattern. As the user starts pulling a vertex on a seam line, the system first constructs a projection plane that passes through the seam line (Figure 16). The mouse cursor position on the screen is projected onto the projection plane, and it is used as a target position for the pulled vertex during subsequent dragging. The system computes the displacement δ_i in the local coordinate frame on the projected plane from the original position v_i to the target position h_i, and moves the corresponding vertices ${u_i}^0$ and ${u_i}^1$ in the 2D mesh in their local coordinate frames by that amount δ_i. These 3D and 2D coordinate frames are defined by the pulled vertex's normal vector and the direction of the seam line. As in the adjustment process after the initial creation, the system updates the 2D mesh by applying Laplacian smoothing and then uses the result to define the new rest edge length for guiding simulation. The system iterates this displacement process with physical simulation until it converges. To achieve a smooth deformation, the system also moves the surrounding vertices in the 2D mesh using the curve manipulation method introduced in Igarashi et al. [2005]. It enlarges the region to be deformed proportional to the displacement of the pulled vertex.

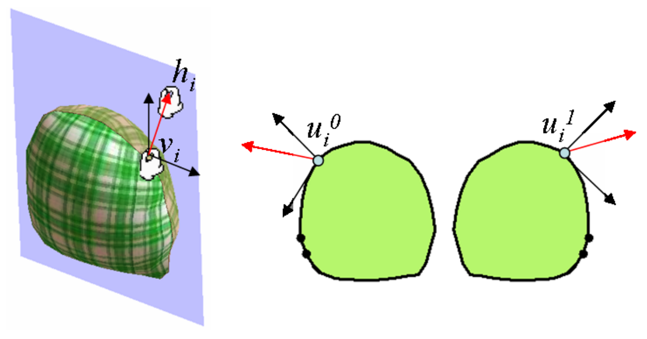


Figure 16: *Pulling a vertex on a seam line.*

Insertion and Deletion of Seam Lines: Insertion of a new seam line is straightforward. The system simply cuts the patch along the added seam line and updates the meshes accordingly. Deletion is more complicated because the merged patch is not necessarily developable. The system applies an approximate flattening operation [Sheffer et al. 2005] to the merged 3D surface to obtain the geometry of the new 2D patch.

4 Results

Plushie is implemented as a Java$^{\text{TM}}$ program. Construction of 2D patterns and a physical simulation run in real-time on a 1.1 GHz Pentium M PC. We designed a couple of plush toys using our system and created a real toy based on the printed pattern. A modeling session typically takes 10-20 minutes and sewing takes 3 -5 hours. Figure 17 shows a plush toy and balloon model designed in our system. It shows that the physical simulation successfully captures the overall shape of the real objects. We interviewed with professional balloon designers and they supported our system, saying that it can significantly reduce the time necessary for designing original balloon.

The user can assign different textures to individual patches (Figure 18). Therefore the user can explore various design possibilities before actuary sewing the real fabric. These models also demonstrate the effectiveness of thin parts.

We ran a small workshop to have novice users try our system. Nine children, 10-14 years old, joined the workshop accompanied by their parents. We gave a brief tutorial at the beginning and had them design their own plush toys using the system. It took about an hour for the design. They then printed the designed pattern and sewed a real toy in approximately 3 hours. Figure 19 shows a couple of plush toys created in the workshop. Participants quickly learned how to use the system and successfully designed the 3D models they wanted, with some help from volunteers. Furthermore, they enjoyed the process. These toys were their own creations and one-of-a-kind designs. Participants also gave us valuable feedback for future improvement. They wanted to have some auxiliary functions

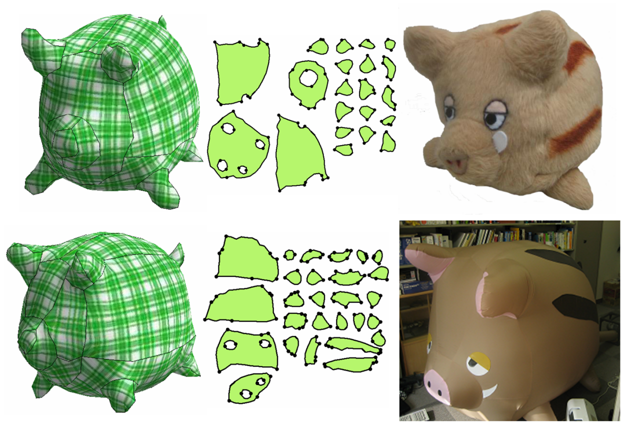


Figure 17: *A plush toy and a balloon designed in our system.*

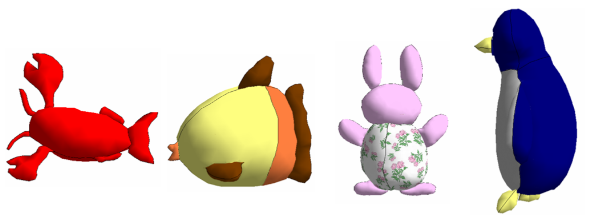


Figure 18: *Example of texture changed. These models have many thin parts.*

such as the ability to design symmetric parts and remove existing parts, but no one complained about the quality of the visual simulation. A perfectly accurate simulation is not necessary because many small variations inevitably occur during the real sewing and stuffing process.

5 Limitations and Future Work

Our 3D pull operation works well for inter-patch seams as shown in Figure 6, but not for intra-patch seams, i.e., darts. If the user pulls a dart line outward in the 3D view, the system tries to move the corresponding patch boundary to the normal direction, which causes a self-intersection within the patch (Figure 20 left). We currently rely on the 2D pull operation to open a dart as in Figure 10, but we would like to allow more intuitive operation on the darts in the 3D view. For example, we want the system to automatically open the darts, for example, when the user pulls a dart outward in the 3D view (Figure 20 right).

We currently do not consider the bending energy in our simulation, so the material is a bit too flexible. Based on the experience of our users, this is not a serious problem for most target materials, but appropriate treatment of bending can extend the applicability of our system to more inflexible materials such as felt and leather. We plan to prepare several predefined material parameters and allow the user to choose the desired setting for each target material.

Apart from adding a miscellaneous utility function to the system, we would like to incorporate more domain knowledge into the system so that an inexperienced user can design more sophisticated patterns. For example, it is necessary to insert several darts into a patch if the user wants to have it swell more. We would like to develop an intelligent interface that allows the user to pull the center of a patch to its normal direction and then automatically insert darts.

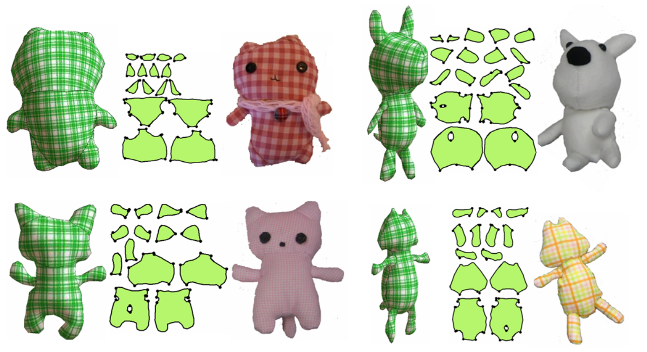


Figure 19: *Example of original plush toys designed and created by children in the workshop.*

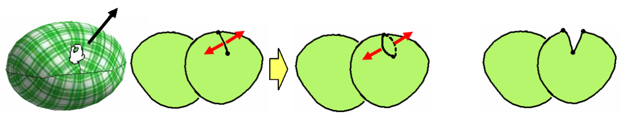


Figure 20: *Limitation of our 3D pulling operation.*

It would be also interesting to provide an intelligent guide to help the sewing process, such as showing the user the order in which they should sew the patches [Agrawala et al. 2003] and providing an estimate of the time required to complete the sewing.

Interactive 3D modeling assisted by concurrent physical simulation can be a powerful tool in many application domains. For example, if one can run an aerodynamic simulation during the interactive design of an airplane model, it might be helpful to intelligently adjust the entire geometry in response to the user's simple deformation operations so that it can actually fly. This kind of interaction would make it easier for designers to pursue aesthetic goals while satisfying engineering constraints. Real-time simulation does require high-performance computing resources, but some meaningful support should be possible by carefully limiting the target task and designing appropriate interfaces as shown in this paper. We hope that our work inspires future work in this direction.

Acknowledgements

We would like to thank Dennis Cosgrove for providing Jalice scenegraph. We also thank Hiromasa Suzuki, Jun Mitani, and Olga Sorkine for their helpful comments. We also thank National Museum of Emerging Science and Innovation in Japan for offering the place to run the workshop and "Quest And Try" Inc. for making the balloon. Finally we thank the user interface research group at the University of Tokyo for helpful discussions.

References

AGRAWALA, M., DOANTAM, P., HEISER, J., HAYMAKER, J., KLINGNER, J., HANRAHAN, P., AND TVERSKY, B. 2003. Designing effective step-by-step assembly instructions. *ACM Transactions on Graphics (Proceedings of SIGGRAPH 2003) 22*, 3, 828–837.

BREEN, D. E., HOUSE, D. H., AND WOZNY, M. J. 1994. Predicting the drape of woven cloth using interacting particles. In *Proceedings of ACM SIGGRAPH 1994*, 365–372.

BRUNO, L., SYLVAIN, P., NICOLAS, R., AND JEROME, M. 2002. Least squares conformal maps for automatic texture atlas generation. In *Proceedings of ACM SIGGRAPH 2002*, 362–371.

CHOI, K. J., AND KO, H. S. 2002. Stable but responsive cloth. In *Proceedings of ACM SIGGRAPH 2002*, 81–97.

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

DESBRUN, M., SCHRÖDER, P., AND BARR, A. 1999. Interactive animation of structured deformable objects. In *Proceedings of Graphics Interface 1999*, 1–8.

GRINSPUN, E., KRISL, P., AND SCHRÖDER, P. 2002. CHARMS: A simple framework for adaptive simulation. In *Proceedings of ACM SIGGRAPH 2002*, 281–290.

IGARASHI, T., AND HUGHES, J. F. 2001. A suggestive interface for 3d drawing. In *Proceedings of 14th Annual Symposium on User Interface Software and Technology (Proceedings of ACM UIST 2001)*, 173–181.

IGARASHI, T., AND HUGHES, J. F. 2002. Clothing manipulation. In *Proceedings of 15th Annual Symposium on User Interface Software and Technology (Proceedings of ACM UIST 2002)*, 91–100.

IGARASHI, T., MATSUOKA, S., AND TANAKA, H. 1999. Teddy: A sketching interface for 3d freeform design. In *Proceedings of ACM SIGGRAPH 1999*, 409–416.

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

JULIUS, D., KRAEVOY, V., AND SHEFFER, A. 2005. D-charts: quasi developable mesh segmentation. *Computer Graphics Forum (Proceedings of Eurographics 2005) 24*, 3, 981–990.

KARPENKO, O. A., AND HUGHES, J. F. 2006. Smoothsketch: 3d free-form shapes from complex sketches. *ACM Transactions on Graphics (Proceedings of SIGGRAPH 2006) 22*, 3, 589–598.

MASRY, M., AND LIPSON, H. 2005. A sketch-based interface for iterative design and analysis of 3d objects. In *Proceedings of Eurographics Workshop on Sketch-Based Interfaces*, 109–118.

MILENKOVIC, V. J. 1999. Rotational polygon containment and minimum enclosure using only robust 2d constructions. *Computational Geometry 13*, 1, 3–19.

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

MORI, Y., AND IGARASHI, T. 2006. Pillow: Interactive pattern design for plush toys. In *DVD publication at SIGGRAPH 2006 Sketches*.

SHATZ, I., TAL, A., AND LEIFMAN, G. 2006. Paper craft models from meshes. *The Visual Computer: International Journal of Computer Graphics (Proceedings of Pacific Graphics 2006) 22*, 9, 825–834.

SHEFFER, A., LÉVY, B., MOGILNITSKY, M., AND BOGOMYAKOV, A. 2005. ABF++: Fast and robust angle based flattening. *ACM Transactions on Graphics 24*, 2, 311–330.