Submission to GMP 2018

Folding Cartons: Interactive Manipulation of Cartons from 2D Layouts

Submission number: 1074

Abstract

Cartons are broadly used in packaging industry and our daily life. While most of traditional packages are axis-aligned cuboids, there is a recent trend of creating paper packages and cartons with more complicated geometry. In this work, we present an interactive tool to manipulate 3D cartons from 2D layouts. Two key components, *automatic folding* and *suggestive editing*, make our system significantly easy to use. The automatic folding part automatically computes the 3D coordinates of vertexes in cartons based on a series of simple rules. Then the suggestive editing part smartly provides a group of geometric constraints, such as vertex merging, shape symmetry, to generate more regular 3D shapes for users to explore. Moreover, users are allowed to edit the shape of 3D model, and refine the 2D layout automatically.

1 Introduction

Cartons have been widely used in packaging industry to deliver various commodities including food items, daily necessities and electronic components. Instead of very basic packaging shapes like cubes, there exist multiple fantastic cartons to package wedding candies or take-away coffee cups. These various designs increase much popularity, not to mention they are environmental friendly due to their recycling and degradability [21].

Cartons are usually designed based on experience and trial-and-error, though recent years there are softwares developed to help designers improve efficiency and productivity. For example, KASEMAKE [1] can help users pick the existing structural designs from database and feed in required basic size and material. Moreover, users can re-import the finished artwork to show the print on the structural designs and fold it into a three-dimensional view in seconds. However, it still costs much work and time to construct 3D model directly from 2D design layout. Sometimes, it is intractable to fold an irregular box layout to the final carton without instructions.

Papercraft problems have attracted a large mount of attentions in computer graphics community for decades. Researchers primarily concerned with the following: (1) computational algorithm and mathematic analysis for folding origami [10, 15, 31]; (2) problems on the folding polygon and unfolding polyhedron [2, 23, 25], and some prooves to show the problem complexity of folding to polyhedron [3, 4, 18]; (3) other specific structure folding problems like Kinetogami which comprises multi-primitive and reconfigurable units folded from a single sheet of paper [9] and applications to show the folding behaviour of paper [30, 14, 22, 26]. Although previous works solved plenty problems in paper folding, they have not considered intelligent construction 3D models given 2D paper layouts. (xuejin:TBD..) In this paper, we focus on the carton folding problem and introduce a shape optimization method.

(xuejin:discuss more about the challenges in this problem. Why is it difficult?) Given a structural layout of a carton only, it is hard to formulate the final state of model mathematically because the information extracted from layout is not enough to represent the shape of corresponding model. Besides, Biedl et al. proved that given a polygon and a set of creases, it is NP-hard to know whether a polyhedron can be obtained by folding along the creases [3], which can illustrate the difficulty of our problem in some way.

Nevertheless it is also important to construct 3D models from layout, on one hand, the layouts of cartons are more reachable for common users, but they can not imagine how the final model is constructed. On the other hand, even users can obtain layouts by unfolding 3D objects, the unfolded layouts are fragile in some parts which are not suitable to fabricate and sometimes are not feasible to fold.

In this paper, shape constrains for carton are proposed to optimize 2D design layout into the corresponding 3D realization. Moreover, an interactive design and exploration framework is developed to allow users visualize the structural layout in a 3D view, which can improve the efficiency a lot when designing a complicated paper package. The contribution of this paper includes the following:

(1) An interactive and exploration system is presented to manipulate the shape of 3D carton. It creates the corresponding 3D realization by initialization and user interaction, and users are allowed to edit the 3D model to explore deformed layouts automatically.

(2) We propose shape constrains represented by a set of points to implement shape optimization. Observed from existing cartons, constrains are summarized including edge length constrain, coplanarity and face rigidity to keep the shape of cartons. Besides, computer-aid detection like vertex merging and symmetry detection are brought to give users suggestions.

2 Related Work

2.1 Paper folding problem

Various types of paper crafts have been studied in the field of computation and mathematics. Origami is the Japanese traditional paper art of making different kind of objects by a single sheet of paper, and has been long studied since 1970s [12].

While researching on origami in the field of computational algorithm and geometric analysis, the simulation system is developed to visualize the folding behaviour of a single piece of paper. Thiel provides a virtual origami system

including the user interface to model folded paper and show animations of folding process [30]. Kishi et al. allows users create and edit the origami properties over the Web [14]. Nimnual et al. presented an application for package folding practices in a virtual space [22]. Although these applications can model folded paper well, they need given parameters to construct the model, and how to align constrains to structural layout by implementing shape optimization is our main concern.

There are also some methods to solve the problem of carton folding. Song et al. modeled foldable objects as tree like multilink objects and used PRMs (probabilistic roadmap methods [13]) to find a sequence of motions to transform some configuration of a foldable object into another configuration [28]. Mullineux et al. provided a simulation of the carton during erection using a constraint-based approach. Both these work required the target state as a premise, while our work aim to generate the target configuration [21].

The complexity of folding to polyhedron problem has also been studied for decades, Lubiw provided an dynamic programming algorithm based on Aleksandrov's theorem to test whether a polygon can be folded into polyhedra which takes $O(n^2)$ time and space [18]. O'Rourke examined three open problems on the subject of folding and unfolding [23]. Biedl et al. has studied in polynomial time to solve the question of when is the graph orthogonally convex polyhedra given a graph, edge length and facial angles, also shown that it's NP-hard to decide whether the graph is orthogonally polyhedra or not [4]. Rather than the given graph, Biedl et al. proved that if given a net along with the dihedral angle at each crease, we can know whether a net can be folded to a polyhedron in polynomial time, but it becomes NP-hard without the angles even adding constrains on orthogonal polyhedron, which results in more difficulties on more complex input [3]. Compared to our desired result, polyhedron is a set of polygons without overlap, nevertheless, our 3D model contains paste faces that needs to be fixed to another panel. These works above justify our problem being hard to solve caused by even more intricate inputs.

2.2 Reconstruction from single line drawings

A line drawing is defined as a 2D projection of object containing its vertexes and edges. Line drawings of three-dimensional objects have long been studied, and the main problem is still in object reconstruction given its projection on two-dimensional planes. Some researchers treat this task as optimization problem. Marill proposed MSDA (Minimize the Standard Deviation of Angles) principle to emulate the interpretation of line drawings as 3D objects [19]. This new criterion is used by many other researchers later. Leclerc et al. combined MSDA with the deviation from planarity as objective terms [16]. Cao et al. added symmetry measure of the objects to get more complicated results [7]. Some other researchers try to solve this problem from the information theoretic point of view. Marill minimized the description length of objects based on the idea that we usually pick the simplest one from infinite possibilities when we see the line drawing [20]. Shoji et al. implemented the principle of minimizing the entropy of angle distribution between line segments using genetic algorithm [27]. Later, split and merge strategy has been used to solve line drawings of complex 3D objects [17, 32].

Different from the input above, ours are expanded structural layout of three-dimensional objects in 2D planes, the final model is constructed based on the folding process instead of mathematical presentation.

2.3 Shape optimization

Multiple algorithms have been proposed to enforce shape constrains, and have been used successfully for interactive tools and physical simulation [5, 11]. Bouaziz et al. unified a large variety of geometric constrain into one optimization framework, and provided simple, and robust implementation [6]. Poranne et al. described an interactive method to manipulate and optimize polyhedral meshed with constrains with a linear-time algorithm [24]. Tang et al. solved constrained equations by Newton-type method in a fast way, and provided an interactive system to model meshed constrained by equalities and inequalities [29]. Deng et al. developed an interactive tool to explore architectural design with shape constrains, and provided an optimization method to enforce hard constraints and soft constraints at the same time [8]. (xuejin:In comparison, we provide....)

The studies mentioned above focus on the architectural design with constraints, and proposed different solutions, while our constrains do not need satisfy properties like fairness. As a result, we implement the shape optimization by the method introduced in [6] for its robustness and simplicity.

3 Overview

Figure 1 shows the overview of our algorithm. As shown in Figure 1(a), the input 2D design layout of a carton consists of a set of cutting edges (solid lines) and folding edges (dashed lines). Given the 2D layout, an undirected graph is built and faces are extracted by finding minimum cycles in the graph, as Figure 1(b) shows. The 2D layout therefore can be represented by a polymesh $\mathcal{L} = (V, E, F)$, where V is the set of vertexes, E is the set of all edges, and F is the set of faces. The edge set $E = E_c \cup E_f$, where E_c is the set of cutting edges, and E_f is the set of folding edges. To build a 3D carton model $\mathcal{M} = (V, E, F)$ from the 2D layout L, while they share the same topology, we compute the 3D coordinates of all vertexes in V.

However, it is not intuitive to analytically define the desired final 3D shape from a 2D layout. One possible way is to detect all the possible geometric constraints such as vertex merging, face parallelism, adjacent face orthogonality, and then to integrate all these possible constraints together to form a large equation system. But the challenge is that these local constraints can not well describe complicated and creative designs. Moreover, there are many ambiguities when detecting these constraints in a 2D layout that consists of many repeated edges and faces. We propose a two-step algorithm based on the observation that humans usually fold edges with a right angle to get a rough shape and then merge vertexes or edges to get a stable 3D carton. First, an initial 3D model (Figure 1(c)) is constructed based on (xuejin:a set of simple rules)a specific angle along each fold edge, as described in Sec. 4.1. The user can then manipulate

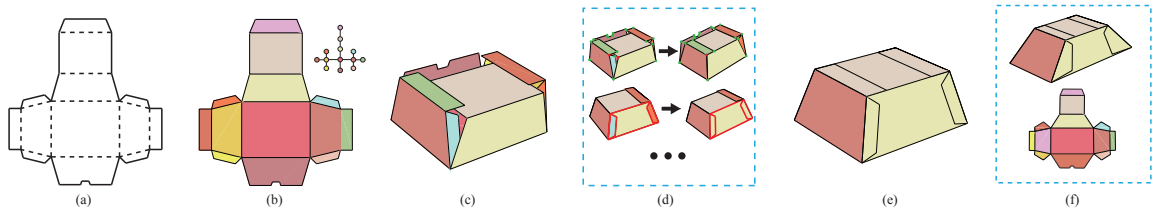


Figure 1: Given a 2D layout (a), we first extract its 2D mesh (b), with different face in different color. By providing each folding edge with a specific angle, we can construct an initial 3D model (c). The final carton model (e) is built through the shape optimization based on the information acquired from user interactions (d). (xuejin:update caption for new figure.) Our system also allows users to manipulate the 3D model, and explores deformable layout by enforcing geometric constraints reversely to the flat mesh (f).

and explore 3D shapes of the canton based on a series of suggestive operations provided by our system. The final model is shown in Figure 1(e).

(xuejin:Modify this overview when you have new figure.) Furthermore, our system allows users modify the final 3D model to the desired shape, and automatically enforce the shape constraints reversely to the flat polymesh to get a deformable layout as shown in Figure 1 (f).

4 Algorithm

In this section, we explain our algorithm in detail. Initially, assume that the 2D layout $\mathcal{L}$ lies on the XOY plane, so that each vertex $\mathbf{v}_i(x_i, y_i, z_i)$ in V has $z_i = 0$. Each face in F has a normal $\mathbf{n}_i = (0, 0, 1)^T$. In the first step of our algorithm, our system automatically folds the original flat mesh into a rough model by assigning a specific angle to each folding edge. In the second step, the carton shape is refined based on a set of potential geometric constraints.

4.1 Shape Initialization

A general process for humans to fold a carton starts from folding each edge by a rough angle and then connecting close vertexes to obtain a box-like shape. Naturally, we first fold the 2D layout by assigning a rotation angle to each pair of adjacent faces to get a rough shape, instead of directly computing the vertex coordinates. Observed from existing data in the Internet, most of the traditional cartons are cuboid for holding files or delivering daily supplies. Although there is a recent trend to design more complicated layouts to attract consumers, the shape of these unusual cartons is similar to boxes as their functions are still packaging commodity. Therefore, we choose $\pi/2$ as the initial value of the rotation angle to each folding edge. First, a face graph of a layout is constructed, as shown in Figure 2 (a). Each face is a node, and there is an edge between two faces if they are connected by a folding edge. The faces in the 2D layout are recursively folded in a breadth-first manner. Starting from the face with the maximal area, our system folds all its adjacent faces by rotating them around their connecting edge by $\pi/2$, and then continues to others. Figure 2 illustrates the folding process of a cuboid carton.

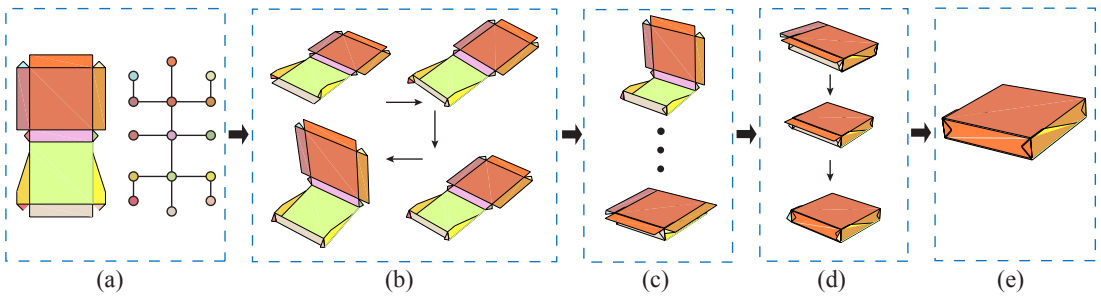


Figure 2: A rough 3D shape is obtained by folding each face by $\pi/2$ in the 2D layout (a) in a breadth-first manner, starting from the face with the maximal area. (b), (c), (d) and (e) are the different folding stages in the initialization step.

Figure 3 shows a group of results generated by simply folding faces by $\pi/2$. Traditional cartons in cuboid shapes could reach an ideal state, as Figure 3 (a) shows. However, many complicated designs still need refinement, such as the four examples in Figure 3 (b). Take the hexagonal box as an example, the 3D model can be improved by simply snapping the small paste faces to its nearby faces. As a result, we provide an suggestive interface by automatically detecting potential geometric modifications in the rough carton shape for users to explore.

4.2 Shape Refinement

Though not perfect, the initial 3D shape provides a good start point for generating the final carton model. To further refine the 3D shape, the 3D coordinates of all the vertexes are then refined based on a set of shape constraints, such as vertex merging, face pasting, and so on. In this step, the coordinate of vertexes are chosen as our objective instead of

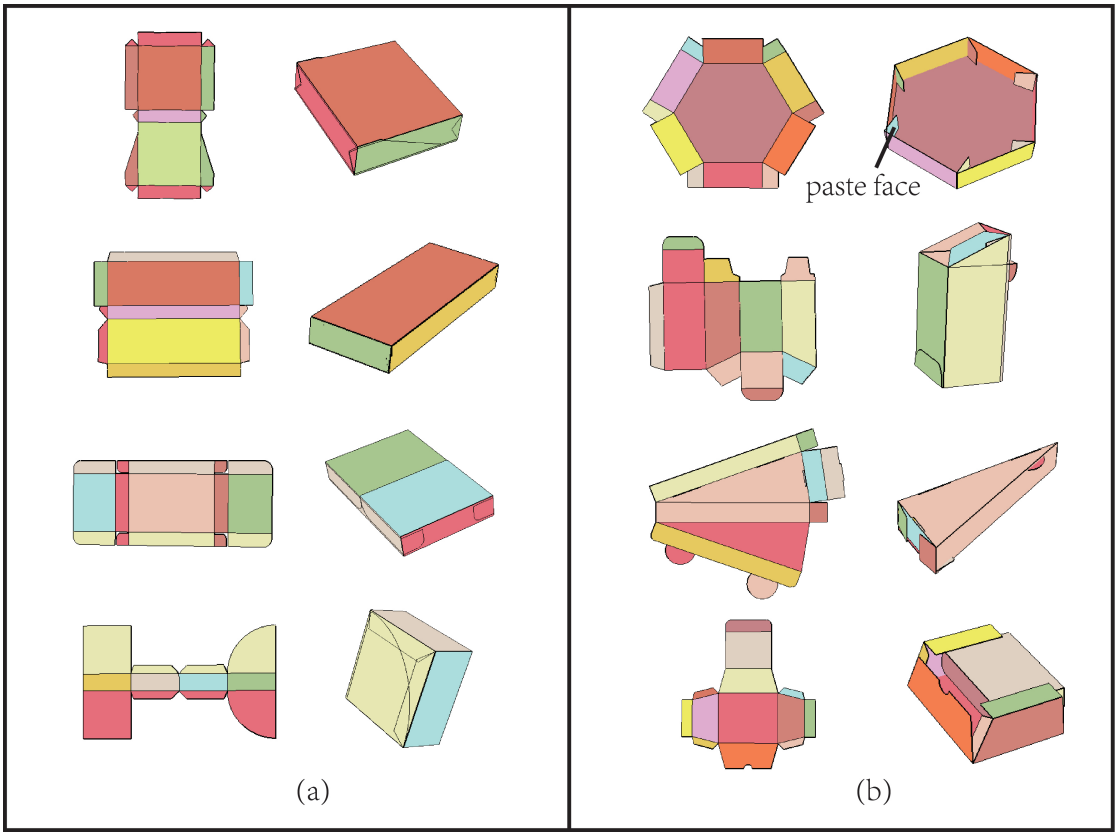


Figure 3: Eight different initialization results. Four initial cartons shown in the second column of (a) can reach the ideal state, the other four cartons need further refine (b).

angles on folding edges, mainly because that the geometric constrains in 3D shapes can be more simply and intuitively represented by 3D vertexes.

A suggestive interface is provided to users for efficiently explore better carton shapes, as described later in Sec. 5. Once the user selects a suggested shape refinement operation, we optimize the 3D shape based on a series of geometric constraints. Given the current mesh whose vertex positions are defined as $\{\hat{\mathbf{v}}_i\}_{i=1}^N$, a new mesh with the same topology but new vertex positions $\{\mathbf{v}_i\}_{i=1}^N$ is constructed. The geometric constraints can be classified into two groups, (xuejin:shape rigidity constraints and shape modification constraints). First, to keep the rigidity of each face, the constraints including edge length constraint, coplanarity, and face rigidity, which are corresponding to similarity constraint and plane constraint described in [6]. Second, once two vertexes or two faces are confirmed to be merged to modify the rough shape, more constraints are added. Each constraint is defined as following.

Face rigidity Each face keeps its shape unchanged during folding. This constraint is defined by keeping the length of each line connecting any pair of points $\mathbf{v}_a, \mathbf{v}_b$ on the face the same.

$$||\mathbf{v}_a - \mathbf{v}_b||^2 = ||\hat{\mathbf{v}}_a - \hat{\mathbf{v}}_b||^2. \tag{1}$$

Coplanarity For each face in $\{f_k\}_{k=1,\ldots,P}$, the coplanarity constraint specifies all vertexes in a face should always lie on a plane. We can compute the sorted eigenvectors $\mathbf{U} = [\mathbf{e}_1, \mathbf{e}_2, \mathbf{e}_3]$ of the 3×3 covariance matrix $\mathbf{C}^T\mathbf{C}$ where $\mathbf{C} = \{\mathbf{v}_{kj}\}_{j=1}^{N_k}$, $\mathbf{v}_{kj}$ is the jth vertex among N_k vertexes in the face. Plane projection can be obtained by removing the last column of $\mathbf{U}$.

If a set of shape modification operations, such as vertex merging and face pasting, are selected, more constraints are added to optimization vertex positions.

Merging vertexes For any two vertexes $\mathbf{v}_p$ and $\mathbf{v}_q$ that are selected to be merged as the same vertex, we have

$$\mathbf{v}_p - \mathbf{v}_q = \mathbf{0}. \tag{2}$$

Face pasting (xuejin:check if this is right.) If two faces f_a and f_b need to snap together, the whole vertexes of these two faces should satisfy coplanarity constrain.

When the above constrains still lead to an ill-posed problem, soft constraints will be added to keep the original positions of the vertexes that are not relative to the shape modification.

Irrelevant vertexes For the vertexes $\{\mathbf{v}_i\}$ are not in the same plane with any vertex for merging or face pasting, they should stay at their original location. We add these soft constraints by adding a small weight w, which is set 0.001 in our experiments to the following equation.

$$\mathbf{v}_i - \hat{\mathbf{v}}_i = 0. \tag{3}$$

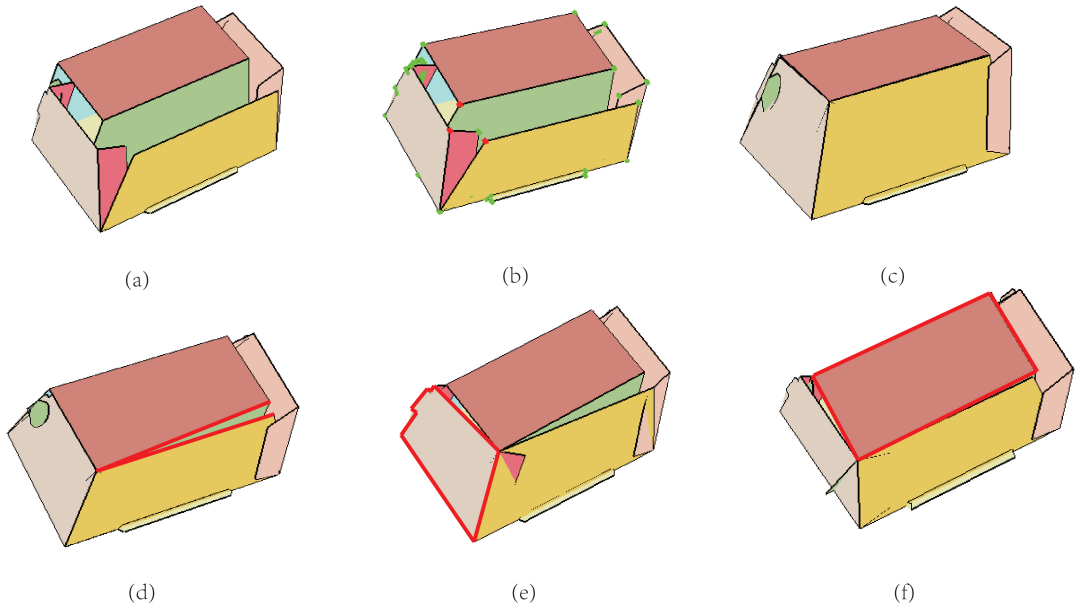


Figure 4: Given an initial shape (a), by merging the three vertexes marked in red (b), basic three shape constrains can lead to the optimization result (c). The bottom three are the results optimized lack of edge length constraint, coplanarity and face rigidity separately, compared to (c), the results can not keep the basic shape well. Without edge length constrain, the two red edges have been stretched (d), the face surrounded by red lines (e) is not a standard plane without coplanarity, and the face surrounded by red lines in (f) has been deformed without face rigidity.

(xuejin:My suggestion on Figure 4: (a)(b) 3D shape before/after vertex merging, highlight the vertexes. (c)(d) 3D shape before/after face pasting.)

Figure 4 illustrates the necessity of basic three constrains. (xuejin:More detailed explaination on the figure. Explain the three constraints in the figure.) When we merge three vertexes circled in red Figure 4 (b) into one location, the optimized model is generated under basic constraints shown in Figure 4. The model shown in Figure 4 (d) (e) and (f) are the results after optimizing without edge length constrain, coplanarity and face rigidity constrain separately. Without edge length constrain, the two red edges have been stretched Figure 4 (d), the face surrounded by red lines Figure 4 (e) is not a standard plane without coplanarity, and the face surrounded by red lines in Figure 4 (f) has been deformed without face rigidity.

5 Suggestive Interface for Shape Modification

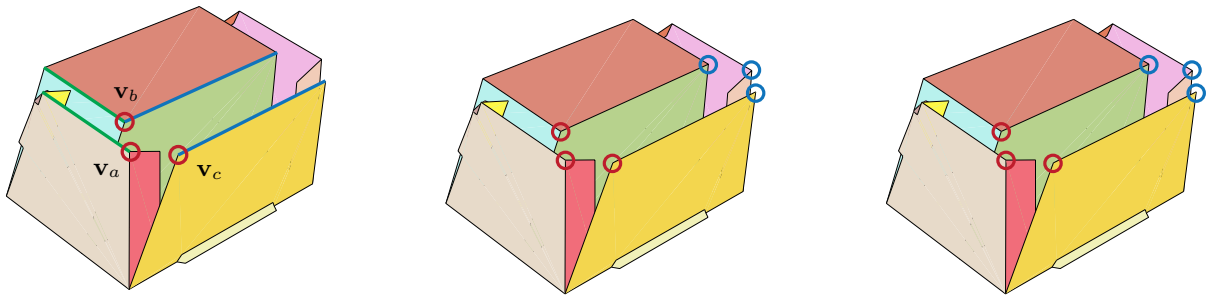
For some creative designs, the rough 3D carton model generated based on our heuristic approach still needs some improvement or the user might edit the carton shape. To assist users to refine the carton model, our system is equipped with a set of smart shape refinement operations, such as automatically detecting vertexes that can be merged, propagating modification to similar groups, and providing suggestions for users to explore. While a user edits the carton model in 3D space, which is more intuitive, our system is capable of revising the 2D layout automatically.

5.1 Suggestive Interaction

Our system automatically detects a group of potential shape constraints to improve the 3D carton model. These guiding constraints are provided to the user for quickly selecting and manipulating the carton shape. In our system, (xuejin:three) types of shape refinement operations, vertex merging, merging propagation, and face pasting, are automatically detected.

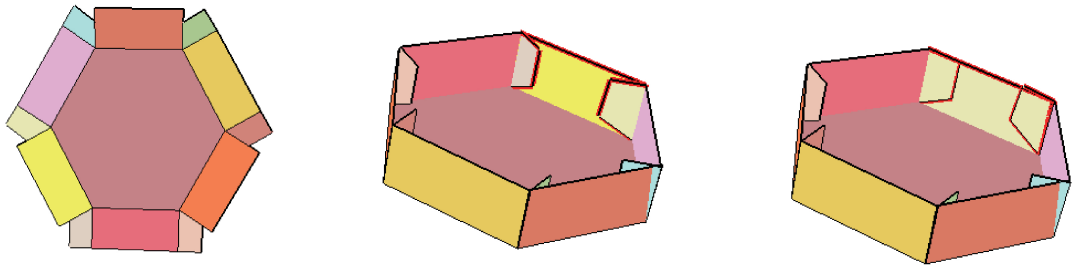
Vertex Merging. For irregular shapes which consist of non-rectangular faces or non-perpendicular adjacent faces, some vertexes are close to each other after the initialization step. A practical carton model can be obtained by simply merging these nearby vertex together if their have coherent edge. Any pair of vertexes $\mathbf{v}_a$, $\mathbf{v}_b$ in different faces are detected as mergeable, if $|\mathbf{v}_a - \mathbf{v}_b| < \epsilon$, and there is an edge connecting to $\mathbf{v}_a$ having the equal length with one edge connecting to $\mathbf{v}_b$, ϵ is a distance threshold and set as 50mm in our experiments. More vertexes can be merged together, as shown in Figure 5(a) by gradually adding one vertex $\mathbf{v}$ each time if $\mathbf{v}$ is mergeable with at least one vertex in current list.

Merging Propagation. Once the user makes a decision to merge a subset of vertexes $\mathbb{V} = \{\mathbf{v}_i\}_{i=1,\dots,K}$ from automatic system suggestions, or manually selection, our system will detect if there exists another subset of vertexes $\mathbb{V}_b$ symmetric to $\mathbb{V}$ and propagate the merging operation to $\mathbb{V}_b$, as shown in Figure 5(b). A subset $\mathbb{V}_b$ is symmetric to $\mathbb{V}_a$ if $|\mathbb{V}_a| = |\mathbb{V}_b|$, and there exists one-to-one correspondence between the vertexes in $\mathbb{V}_a$ and $\mathbb{V}_b$. Here, $|\mathbb{V}|$ is the number of vertexes in a vertex subset $\mathbb{V}$. Two vertexes $\mathbf{v}_a$ and $\mathbf{v}_b$ are defined as corresponding vertexes if they have the same number of edges and one-to-one correspondence can be found between the edge lengths.



(a) Vertex Merging (b) Merging Propagation (c) Symmetry Detection

Figure 5: Vertex merging detection is well illustrated by (a), $\mathbf{v}_a$ and $\mathbf{v}_b$ may have the need to merge cause $|\mathbf{v}_a - \mathbf{v}_b| < \epsilon$, and one edge connecting to $\mathbf{v}_a$ marked as green has the same length with one edge connecting to $\mathbf{v}_b$ also marked as green. $\mathbf{v}_c$ can also be considered that need to merge with $\mathbf{v}_a$ and $\mathbf{v}_b$ as $|\mathbf{v}_c - \mathbf{v}_a(\mathbf{v}_b)| < \epsilon$ and one of incident edges of $\mathbf{v}_c$ marked as blue has the same length as $\mathbf{v}_a(\mathbf{v}_b)$'s also marked as blue. While users select three vertexes circled in red (b), our system can detect symmetric vertexes circled in blue automatically. (xuejin:show more possible suggestions in this figure. Make sure the font in consistent with the caption).



(a) (b) (c)

Figure 6: The initialization result (b) of hexagonal box layout (a) can not reach an ideal state by merging vertexes, hence our system allow users to select merging faces which are bounded by red lines (b) and enforce planarity to the model, the result is shown by assigning same color to the coplane faces (c).

Face Pasting. (xuejin:ToBeDone) Merging vertexes sometimes are not enough to generate desired model caused by the paste faces as shown in Figure 6 (b). A pair of faces f_a and f_b is considered mergable if their normals $\mathbf{n}_a$ and $\mathbf{n}_b$ satisfy $\mathbf{n}_a \cdot \mathbf{n}_b > 0.5$, and for each vertex in $\{\mathbf{v}_{ai}\}_{i=1}^{N_a}$, $|\mathbf{v}_{ai} - \mathbf{v}_c^b| < \epsilon_d$, where $\mathbf{v}_c^b$ is the centroid of face f_b. When involving multiple faces, the detection strategy is same with vertex merging. (Figure needs modification.)

5.2 2D Layout Refinement from 3D Editing

(xuejin:Explain when modify the 3D model, how to change the 2D layout.) In order to explore more design layouts, our system also allows users edit 3D models by dragging the edges into desired place, and enforce the geometric constraints similar to those described in Sec. 4.2. The edge length constraint and face rigidity are still the same, we expanded coplanarity to all the vertexes in the layout as they should lie on the same plane. The Irrelevant vertexes which exclude the two vertexes lie on the moving edge stay as close to their original location as well. Figure 7 shows one example of 3D editing.

6 Results and Discussion

SS: Any need to list the number of steps we take to generate final model?

The step of shape optimization in our paper can be best demonstrated on the model Figure 8, through initialization, we can have a rough concept of the final model, and by user interaction including selecting vertexes merged and faces in the same plane, we can finally generate a pleasing model from a given structural layout.

We show a number of 3D models generated by our system Figure 10, and most of the cases can have an ideal result after initialization because of the cuboid shape as Figure 10 (a). Through interaction, users can folding more complicated cartons to ideal shapes as Figure 10 (b).

Add table for number of operation steps

However, the initialization will not generate a pleasing result on some complicated cases, and leads to the difficult interaction to get the final model. As for Figure 9, to have a better corresponding model, users need to take at least ten steps to get to Figure 9(c).

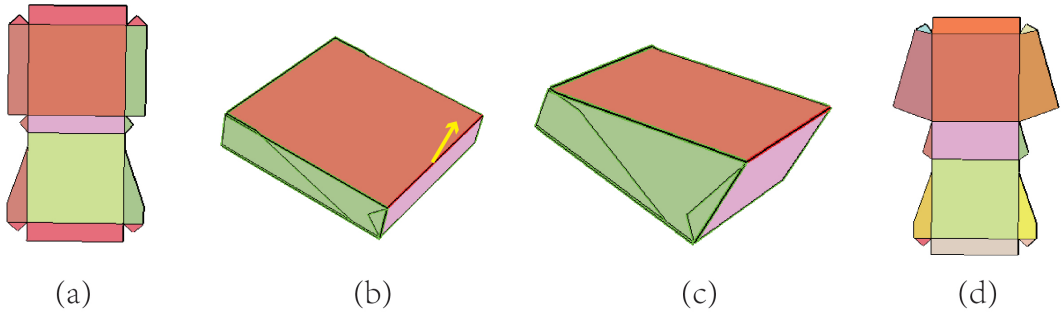


Figure 7: A standard cuboid carton (b) can be generated by the layout (a). Users are allowed to edit the shape of 3D model by dragging the edges, then the model will be deformed to a novel carton (c), by enforce the constrains to layout, our system can generate a deformable layout automatically as (d) shows.

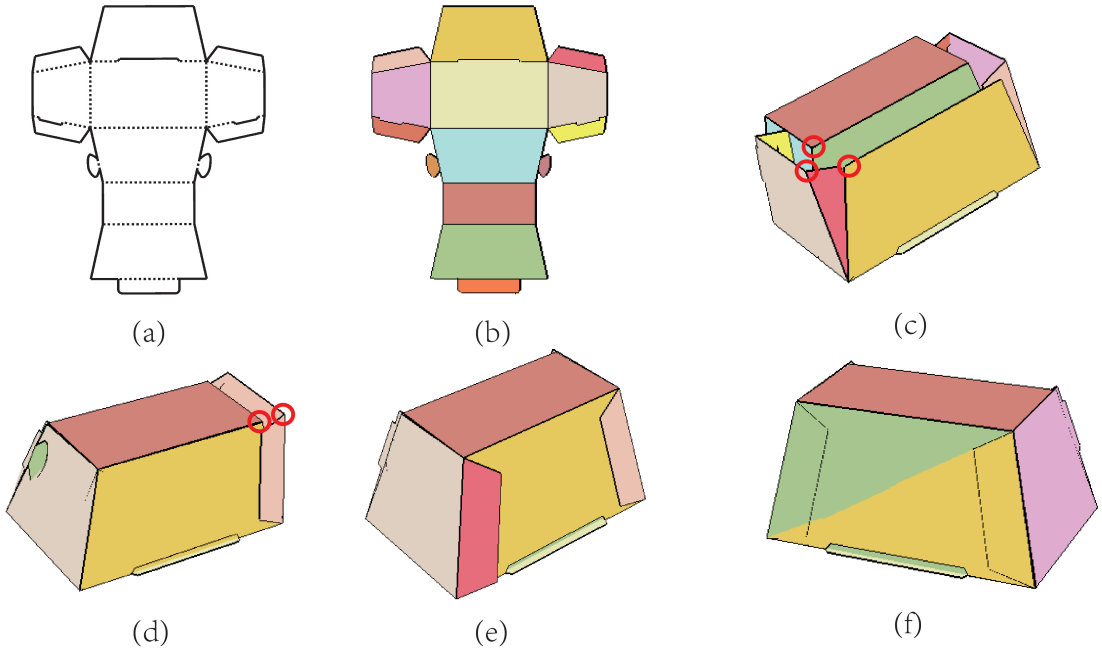


Figure 8: Structural layout (a) is used to generate flat polymesh (b), and then initialize to a rough model (c), after merge the three vertexes in red circles (c) and the two vertexes circled red (d), the model is almost closed with two paste faces in red and pink outside the carton (e), finally through selecting these faces are in the same plane of the yellow face as surface, we can have the final model (f).

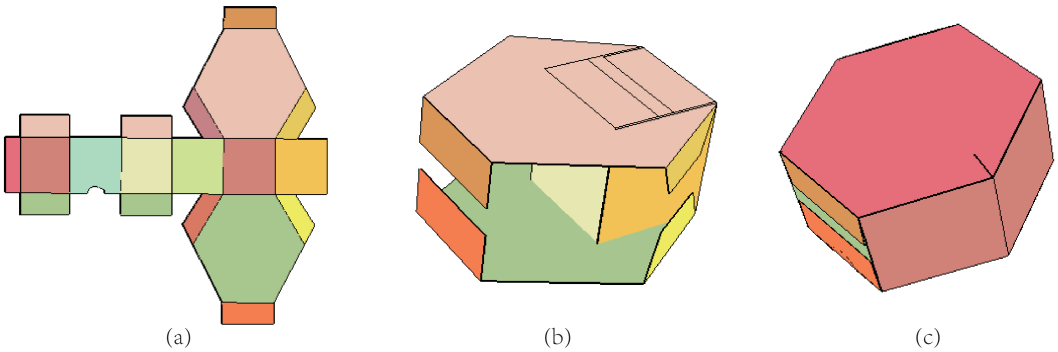


Figure 9: Given a layout as (a), our system can generate an initialized result as (b), and through at least ten steps including select vertexes need to be the same place and faces need to be coplane , users can get the final model as (c).

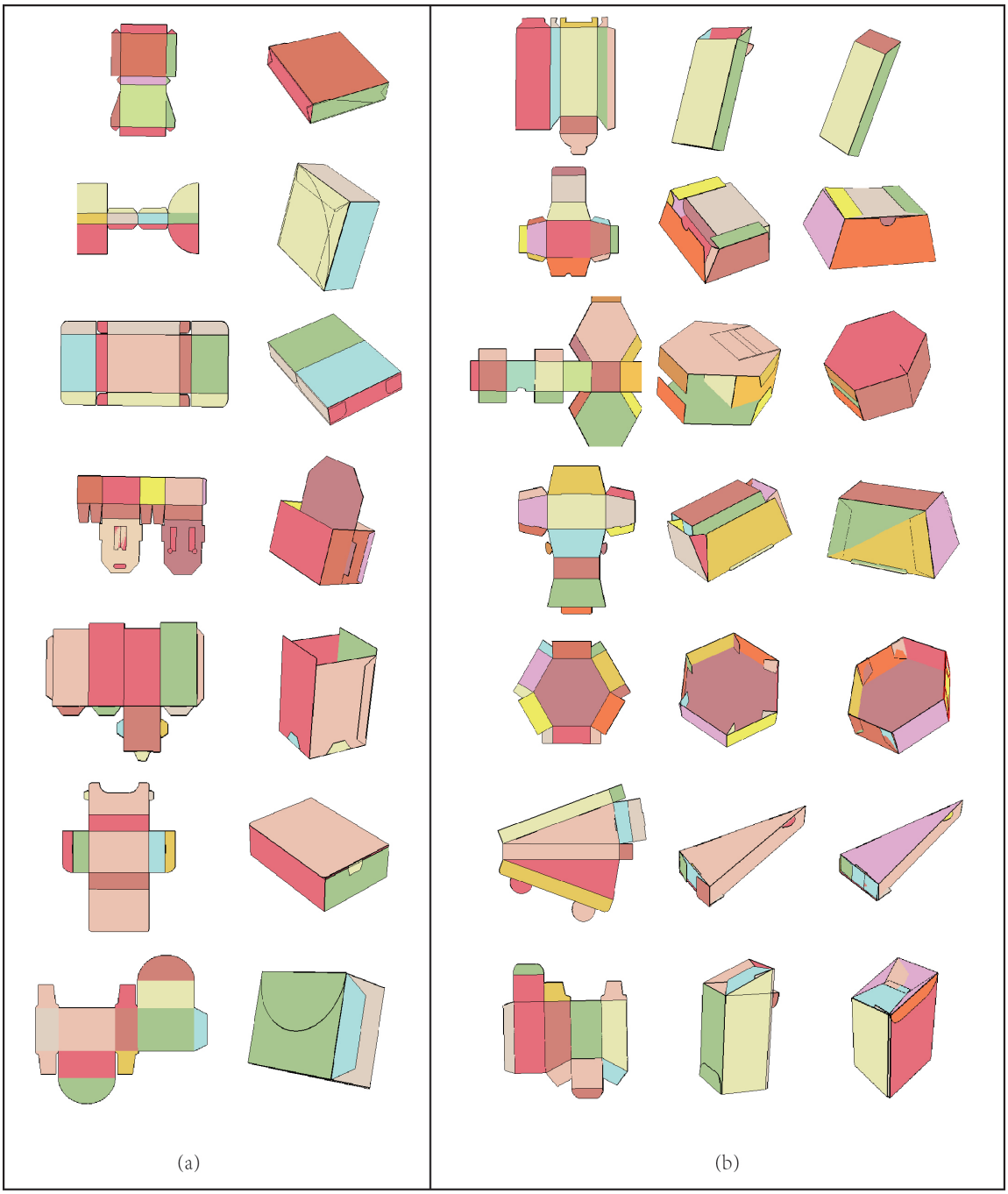


Figure 10: More results, part of cartons can be initialized without refining (a), and with interaction, users can manipulate more complicated cartons (b). The first and second column in (a) and (b) are the flat mesh and initialization result of cartons, and the last column in (b) shows the model after interaction.

7 Conclusion and Future Work

In this paper, we present an interactive modeling system to construct a 3D carton model from 2D expanded structural layout. A series of shape constrains of cartons are provided for smartly optimizing and editing the initial 3D model, which assists users on producing the ideal model efficiently and productively. The direction of future work is adding more operations to our system, including adding the paste face through stability detection or design the surface pattern of a carton either in 2D or 3D.

References

[1] AG/CAD. Kasemake packaging design software. http://www.agcad.co.uk/software/software.html.

[2] M. Bern, E. D. Demaine, D. Eppstein, E. Kuo, A. Mantler, and J. Snoeyink. Ununfoldable polyhedra with convex faces. *Comput. Geom. Theory Appl.*, 24(2):51–62, Feb. 2003.

[3] T. Biedl, A. Lubiw, and J. Sun. When can a net fold to a polyhedron? *Comput. Geom. Theory Appl.*, 31(3):207–218, June 2005.

[4] T. C. Biedl and B. GenÃ§. When can a graph form an orthogonal polyhedron? pages 100–102, 2004.

[5] M. Botsch, M. Pauly, M. Gross, and L. Kobbelt. Primo: Coupled prisms for intuitive surface modeling. In *Proceedings of the Fourth Eurographics Symposium on Geometry Processing*, SGP '06, pages 11–20, Aire-la-Ville, Switzerland, Switzerland, 2006. Eurographics Association.

[6] S. Bouaziz, M. Deuss, Y. Schwartzburg, T. Weise, and M. Pauly. Shape-up: Shaping discrete geometry with projections. *Comput. Graph. Forum*, 31(5):1657–1667, Aug. 2012.

[7] L. Cao, J. Liu, and X. Tang. 3d object reconstruction from a single 2d line drawing without hidden lines. In *Proceedings of the Tenth IEEE International Conference on Computer Vision (ICCV'05) Volume 1 - Volume 01*, ICCV '05, pages 272–277, Washington, DC, USA, 2005. IEEE Computer Society.

[8] B. Deng, S. Bouaziz, M. Deuss, A. Kaspar, Y. Schwartzburg, and M. Pauly. Interactive design exploration for constrained meshes. *Computer-Aided Design*, 61:13 – 23, 2015.

[9] W. Gao, K. Ramani, R. J. Cipra, and T. Siegmund. Kinetogami: A reconfigurable, combinatorial, and printable sheet folding. *Journal of Mechanical Design*, 135(11):111009, 2013.

[10] T. Ida, H. Takahashi, M. Marin, and F. Ghourabi. Modeling origami for computational construction and beyond. In *Proceedings of the 2007 International Conference on Computational Science and Its Applications - Volume Part II*, ICCSA'07, pages 653–665, Berlin, Heidelberg, 2007. Springer-Verlag.

[11] T. Igarashi, T. Moscovich, and J. F. Hughes. As-rigid-as-possible shape manipulation. In *ACM SIGGRAPH 2005 Papers*, SIGGRAPH '05, pages 1134–1141, New York, NY, USA, 2005. ACM.

[12] T. Kanade. A theory of origami world. *Artificial Intelligence*, 13(3):279 – 311, 1980.

[13] L. Kavraki, P. Svestka, J. Latombe, and M. Overmars. Probabilistic roadmaps for path planning in high-dimensional configuration spaces. Technical report, Stanford, CA, USA, 1994.

[14] N. Kishi and Y. Fujii. Origami, folding paper over the web. In *Proceedings of the Third Asian Pacific Computer and Human Interaction*, APCHI '98, pages 337–, Washington, DC, USA, 1998. IEEE Computer Society.

[15] R. J. Lang. A computational algorithm for origami design. In *Proceedings of the Twelfth Annual Symposium on Computational Geometry*, SCG '96, pages 98–105, New York, NY, USA, 1996. ACM.

[16] Y. G. Leclerc and M. A. Fischler. An optimization-based approach to the interpretation of single line drawings as 3d wire frames. *International Journal of Computer Vision*, 9(2):113–136, 1992.

[17] J. Liu, Y. Chen, and X. Tang. Decomposition of complex line drawings with hidden lines for 3d planar-faced manifold object reconstruction. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 33:3–15, 2010.

[18] A. Lubiw. When can a polygon fold to a polytope? *Dept.comput.sci.smith College*, 1996.

[19] T. Marill. Emulating the human interpretation of line-drawings as three-dimensional objects. *Int. J. Comput. Vision*, 6(2):147–161, June 1991.

[20] T. Marill. Why do we see three-dimensional objects? 1992.

[21] G. Mullineux and J. Matthews. Constraint-based simulation of carton folding operations. *Comput. Aided Des.*, 42(3):257–265, Mar. 2010.

[22] R. Nimnual and S. Suksakulchai. Virtual reality for packaging folding practice. In *International Conference on Control, Automation and Systems*, pages 1011–1014, 2007.

[23] J. O'Rourke. Folding and unfolding in computational geometry. In *Revised Papers from the Japanese Conference on Discrete and Computational Geometry*, JCDCG '98, pages 258–266, London, UK, UK, 2000. Springer-Verlag.

[24] R. Poranne, E. Ovreiu, and C. Gotsman. Interactive planarization and optimization of 3d meshes. *Computer Graphics Forum*, 32(1):152–163, 2013.

[25] J. O. Rourke. Unfolding polyhedra. *Plus Magazine*, 2008.

[26] H. Shimanuki, J. Kato, and T. Watanabe. Construction of 3-d paper-made objects from crease patterns. In *Iapr Conference on Machine Vision Applications*, pages 35–38, 2009.

[27] K. Shoji, K. Kato, and F. Toyama. 3-d interpretation of single line drawings based on entropy minimization principle. 2, 2001.

[28] G. Song and N. M. Amato. A motion planning approach to folding: From paper craft to protein folding. Technical report, College Station, TX, USA, 2000.

[29] C. Tang, X. Sun, A. Gomes, J. Wallner, and H. Pottmann. Form-finding with polyhedral meshes made simple. *ACM Trans. Graph.*, 33(4):70:1–70:9, July 2014.

[30] J. M. Thiel. Interactive manipulation of virtual folded paper. Master's thesis, UBC, 1998.

[31] Z. Xi and J.-M. Lien. Folding rigid origami with closure constraints. In *International Design and Engineering Technical Conferences & Computers and Information in Engineering Conference (IDETC/CIE)*, Buffalo, NY, Aug. 2014. ASME.

[32] C. Zou, H. Yang, and J. Liu. Separation of line drawings based on split faces for 3d object reconstruction. *2014 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 00:692–699, 2014.