Sketching Folds: Developable Surfaces from Non-Planar Silhouettes

Amaury Jung, Stefanie Hahmann, Damien Rohmer, Antoine Begault, Laurence Boissieux, Marie-Paule Cani

Sketching Folds: Developable Surfaces from Non-Planar Silhouettes

Amaury JUNG, Stefanie HAHMANN
Univ. Grenoble-Alpes, CNRS (Laboratoire Jean Kuntzmann) and Inria, France
Damien ROHMER
Univ. Grenoble-Alpes, CNRS (Laboratoire Jean Kuntzmann), Inria, CPE Lyon, France
Antoine BEGAULT
Univ. Grenoble-Alpes, CNRS (Laboratoire Jean Kuntzmann) and Inria, France
Laurence BOISSIEUX
Inria, France
Marie-Paule CANI
Univ. Grenoble-Alpes, CNRS (Laboratoire Jean Kuntzmann) and Inria, France

We present the first sketch-based modeling method for developable surfaces with pre-designed folds, such as garments or leather products. The main challenge we address for building folded surfaces from sketches is that silhouette strokes on the sketch correspond to discontinuous sets of non-planar curves on the 3D model. We introduce a new zippering algorithm for progressively identifying silhouette edges on the model and tying them to silhouette strokes. Our solution ensures that the strokes are fully covered and optimally sampled by the model. This new method, interleaved with developability optimization steps, is implemented in a multi-view sketching system where the user can sketch the contours of internal folds in addition to the usual silhouettes, borders and seam-lines. All strokes are interpreted as hard constraints, while developability is only optimized. The developability error map we provide then enables users to add local seams or darts where needed and progressively improve their design. This makes our method robust even to coarse input, for which no fully developable solution exists.

Categories and Subject Descriptors: I.3.5 [**Computer Graphics**]: Computational Geometry and Object Modeling

General Terms: Design, Algorithms

Additional Key Words and Phrases: sketch-based modeling, developable surfaces, folds generation, non-planar silhouettes

1. INTRODUCTION

Cloth and leather products have folds. Their style, elegance and dynamism are the result of careful choice of the location and size of these folds, many of which are designed to remain permanently, even when the product is at rest. Therefore, considering folds is mandatory during design of garments and of other fashion accessories such as boots, bags or hats.

In real life, specific skills and knowledge are required for creating a set of desired folds when prototyping a fashion product with developable material such as cloth or leather. This is the task of professional pattern-makers, who create the first real prototypes from one or two sketches depicting the product. These sketches typically represent products with non-planar silhouette curves, and include the contours of all desired folds, as depicted in Figures 1 and 2. In this work, we call *silhouette curve* the piecewise 3D curve that corresponds to the silhouette from some viewing direction; note that these curves can be discontinuous and non-planar, such as the yellow curve in Figure 1(b) which corresponds to the left silhouette in the front view. The goal of this work is to provide a digital system enabling to automate the stage of generating a 3D model and a set of 2D patterns from the usual design sketches.

Several sketch-based systems were already proposed for modeling 3D garments or other developable surfaces from silhouettes and seam lines, generally over-sketched on a 3D mannequin. 2D patterns are then generated by flattening each surface panel. However, these systems only capture simple garments with planar silhouettes, i.e. silhouette curves restricted to lie in the image plane. Recent techniques enable to progressively refine a model by over-sketching features on either the 2D patterns or directly on the 3D view of the garment, simulated under gravity in real-time [Umetani et al. 2011]. This greatly improves the interactive design experience, when finalizing a product. However, none of these methods can be used to design folded products from sketches: this is the problem we are tackling here.

In this work, we propose the first sketch-based modeling system for general, folded products made from developable material. This requires facing a major challenge: points on the silhouettes of folded products typically form sets of disconnected non-planar curves on the 3D model, as depicted on Figure 1 (highlighted as yellow lines). These points need to be identified in order to be tied to the sketched silhouettes. Meanwhile, developability needs to be optimized and pre-designed internal folds to be taken into account. Our technical contribution is therefore three-fold: firstly, we introduce a simple, sliding constraint paradigm for progressively identifying the location of silhouette points while optimizing for developability. Secondly, we propose a zippering algorithm ensuring that each silhouette stroke always remains exactly covered and optimally sampled by the model after each silhouette matching step. Lastly, the method is complemented by a new formulation for folded developable surfaces that ensures that pre-designed folds will not be flattened out through developability optimizations.

Our solution is implemented in a multi-view sketching system. This prototype enables us to show that complex garments with multiple folds and the associated 2D patterns can be automatically generated from standard input, e.g. from front and side views depicting silhouettes, fold contours, borders and seam lines. While the input sketches are exactly reconstructed, we only optimize developability: therefore, sketches from which exact developability cannot be achieved are robustly handled. Furthermore, our approach

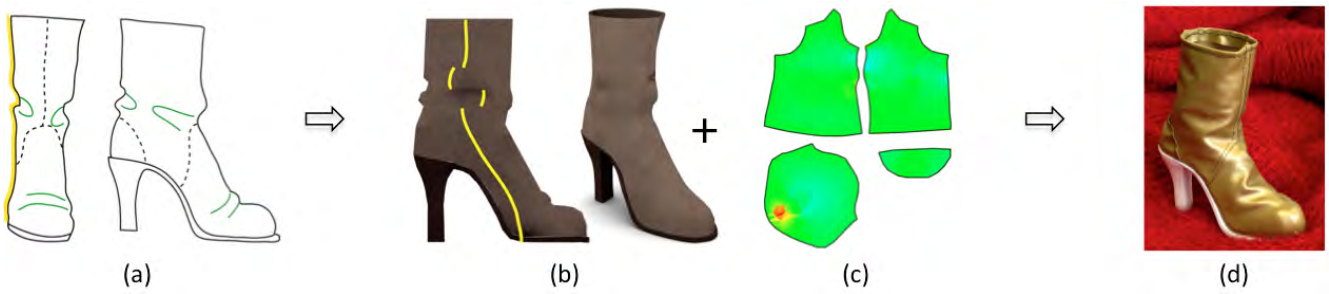

Fig. 1. Our method infers quasi-developable surfaces from multi-view sketches with pre-designed folds. It handles non-planar silhouettes: points on the left silhouette, highlighted by a yellow line, form a non-continuous curve of non-constant depth on the model. The system outputs a 3D model (b) and the 2D patterns (c) enabling to manufacture a real prototype. The red part on the developability error map (c) corresponds to the tip of the boot. Photo of a real leather boot (d), manufactured from the patterns.

provides a developability error map that enables users to progressively improve their design by locally adding seams or darts in non-developable regions. In addition to easing the modeling of plausible virtual garments, our method can serve as a starting point for accelerating the design of real products, by reducing the expensive and time-consuming trial and error loop with real material.

2. RELATED WORK

Cloth and leather products, such as those used for clothing and fashion accessories, are made by sewing together flat panels of material. Their surface is therefore quasi-developable, except in pre-distorted regions such as at the tip of a boot where leather has been heated to give it a rounded shape. Since pieces of flat material tend to bend rather than stretch or compress, they lead to specific, folded shapes. Being able to design these folds is extremely important, since they are typically used for styling the product (see Figure 2).

2.1 Developable surfaces

Developable surfaces are the sub-class of ruled surfaces that have a constant normal vector along each ruling. They include cylindrical, conical and planar parts, as well as tangential surfaces generated by tangent vectors along a 3D curve. Developable surfaces are also characterized by their zero Gaussian curvature, which is equivalent to having the sum of angles around each vertex equal to 2π for meshes. Therefore, they unfold into a 2D shape with no distortion.

Direct modeling of developable surfaces has been extensively studied. Possible classes of input include tensorial patches [Pottmann and Farin 1995], 3D contour curves [Frey 2004; Rose et al. 2007], contour strips [Tang and Wang 2005], 3D positional constraints [Peternell 2004], coarse quadrilateral meshes [Liu et al. 2006], geodesic curves [Bo and Wang 2007] or folding and angular constraints [Solomon et al. 2012]. However, none of these primitives allows the input of silhouette constraints.

An alternative approach for developable surface generation is to progressively optimize developability of a 3D mesh. When the input is already good, an effective solution is to minimize the angular defect around each vertex [Wang and Tang 2004; Wang 2008; Tang and Chen 2009]. Otherwise, the fact that the Gauss map of a developable surface is only 1D can be used as optimization criterion, by making local vertex neighborhoods converge to the best conical approximation [Decaudin et al. 2006]. Our work builds on this criterion, since it is well adapted to folded surfaces. We however introduce a new, linear formulation for efficiently solving the problem. Since surfaces generated after optimization are not exactly developable, a robust method is needed for unfolding surface panels: in this work we rely on ABF++ [Sheffer et al. 2005], a method that computes a conformal shear-minimizing mesh parameterization.

Fig. 2. Design sketches and real products.

2.2 Modeling pipelines for cloth products

Standard digital modeling tools use physical simulation to generate 3D garments from user-specified 2D patterns [Protopsaltou et al. 2002; Volino et al. 2005; CLO3D]. Recent methods, which are already spreading in the industry, speed up the design loop by

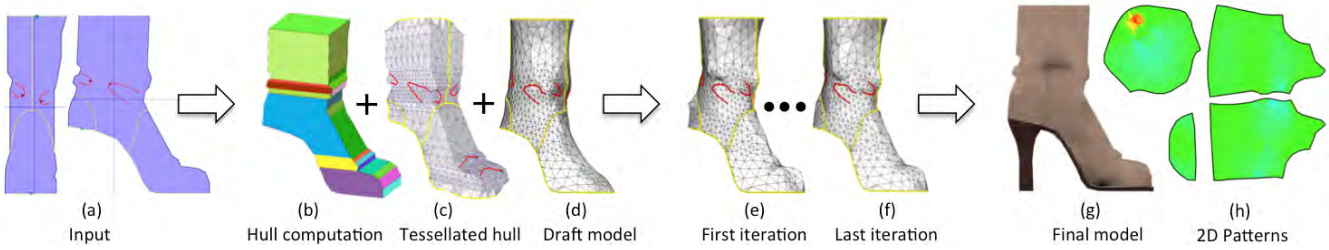

Fig. 3. Processing pipeline: Input (a); Initialization (b,c,d); Iterative optimization (e)...(f); Results (g,h).

enabling direct interaction with both the patterns and the 3D surface [Harmon et al. 2011; Umetani et al. 2011; Marvelous Designer ; Optitex]. Garments can also be distorted and transferred to characters of different morphologies [Brouet et al. 2012]. However, providing an initial set of 2D patterns for a new design still requires expert knowledge, which is not commonplace among artists.

Sketch-based modeling inverts this design loop: it can be used to generate a virtual garment from its contour. Current systems are limited to sketched over a 3D mannequin model [Decaudin et al. 2006; Turquin et al. 2007; Robson et al. 2011]. 3D is either inferred by propagating offset distances [Turquin et al. 2007] or by taking local tightness into account [Robson et al. 2011], while developability can be optimized, leading to the automatic computation of 2D patterns [Decaudin et al. 2006]. The design of closed cloth objects such as plush toys, for which using mannequins would not be appropriate, was achieved using a progressive sketching approach, where parts are added from different view-points [Mori and Igarashi 2007]. However, all these methods interpret silhouettes as flat curves, lying in a plane perpendicular to the viewing direction. Therefore, sketches representing folded surfaces such as in Figure 1 cannot be processed.

Multi-view sketching, where designers specify an object through two or three axis-aligned sketches, was successfully used for various classes of models [Bae et al. 2008; Schmidt et al. 2009; Rivers et al. 2010], but never extended to developable surfaces. Our solution uses the method in [Rivers et al. 2010] as a pre-process.

2.3 Folds design

Design sketches of developable products often depict internal folds (see Figure 2). Taking the folds into account while preserving developabiliy has been an issue for all previous garment modeling techniques. A few solutions were developed for the explicit user-control of folds, either on a 3D surface [Singh and Fiume 1998; Cutler et al. 2007] or on a sketch [Turquin et al. 2007; Zhu et al. 2013]. These solutions however model folds as bumps, which spoils local developability. Other approaches automatically add plausible folds through a post-process, using procedural modeling [Decaudin et al. 2006] or physically based simulation [Rose et al. 2007], but at the price of control. Lastly, developable folding has been investigated for both sharp folds [Kilian et al. 2008] and smooth wrinkling, with inputs either provided by videos [Popa et al. 2009], by physical simulation [Müller and Chentanez 2010; Rohmer et al. 2010], or via machine learning techniques [Wang et al. 2010]. However, none of these approaches can generate folds guaranteed to match the silhouettes on a design sketch. This is one of the challenges we are tackling in this work.

3. THE SLIDING CONSTRAINTS PARADIGM

3.1 Problem setting

Our goal is to generate quasi-developable models, such as garments or fashion accessories, from standard, multi-view design sketches. These sketches typically depict front and side orthogonal projections of a model[1], with an additional top view in some cases. We are looking for a fully automatic method, which is able to compute a 3D model which comprises the four types of features usually represented on such sketches: *Border lines* representing the external boundaries of an open surface; *Seam lines* where individual surface panels are sewn together; *Fold lines* depicting the contours of a fold, a representation more popular in design sketches than the central curve used in [Turquin et al. 2007]; and *Silhouette strokes* delimiting the 2D projection of the sketched model. These features, which can each be drawn on either one or two views, lead to different types of constraints (see also Figure 4):

Border and seam lines correspond to either 2D (if sketched on a single view) or 3D (if sketched on two views) positional constraints for the edges of surface panels.

Fold lines bring orientation constraints for the surface normals at specific 2D (single view) or 3D (two views) positions.

Silhouette strokes generate projective constraints: along them, at least one point of the model should project onto the silhouette, and no other part should protrude more. The set of points on a silhouette typically forms several disconnected curve segments when depicted on the object (Figure 1(b)). If the object is locally flat, this set may include 2D regions.

The surface we are looking for has to exactly meet these constraints while being an optimal solution in terms of developability. Exact developablility is not among our goals: Indeed, there is no guarantee that such a solution exists for a given set of input sketches. Moreover, design with developable material may make use of a few rounded parts such as at the tip of a boot, as already mentioned. Therefore, not requesting exact developability makes our approach robust and general. We however wish to output an error map enabling users to improve developability, if desired, by editing the model or by adding extra seam lines.

3.2 Handling silhouettes through sliding constraints

The specific challenge we are facing is to combine developability enforcement with the constraints of different nature that we just have listed. While positional and orientation constraints can be taken into account, with some effort, within a developability op-

[1] In this work we are not considering perspective drawings.

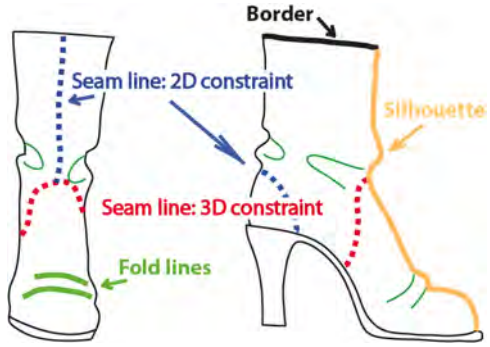

Fig. 4. Typical design sketch with feature lines such as borders, silhouettes, fold and seam lines. Notice that some of the lines are visible in both sketches (red). These lines imply 3D positional constraints in our method, since the corresponding lines on the 3D model have to fit both views of the sketches, see Section 3.3 . Feature lines which are only visible in one sketch are 2D constraints, since the corresponding line on the 3D model have only to correspond to a 2D view.

timization process, projective silhouette constraints are more difficult to handle, as they are not associated to specific model points.

Our first contribution is to introduce the concept of **sliding constraints** to model the non-planar silhouette curves: when the model deforms towards a developable model, the sets of points lying on its silhouettes are allowed to slide over the surface, and to change of topology (curve segments possibly appearing, disappearing, splitting or merging), as it occurs to the silhouette of a real surface when it deforms. To enable this, we optimize developability interleaved with silhouette matching progressively rather than in a single step. This leads us to the processing pipeline described next.

3.3 Processing pipeline

The sliding constraints paradigm we just described requires solving the problem through an iterative optimization process, also summarized in Figure 3:

(1) Initialization: Generate a draft 3D mesh with a feature-aware tessellation from the multi-view sketches.
(2) Optimization loop: While developability can be improved:
  (a) Optimize Developability
  (b) Match Silhouettes

At the end of this loop, the 3D model together with the corresponding set of 2D patterns and a developability error map are generated.

Initialization: We first generate a visual hull from silhouette and border constraints using [Rivers et al. 2010]. A set of planar faces are computed by extruding silhouettes along orthogonal planes (Figure 3(b)). The volume is then meshed according to the internal features (seams and folds), to allow subsequent accurate feature reconstruction: features drawn on a single sketch are projected onto the appropriate faces of the visual hull while those drawn on two sketches are first reconstructed as a 3D curve. This is done as follows: Without loss of generality, let (V_i) be the list of vertices along a line drawn in a front view and (E_j) be the list of edges along the same line in a side view. Our goal is to compute the best mapping from the (V_i) to the (E_j), where mapping a vertex to an edge means that the vertex lies along that edge in the side view (which gives it the missing depth information). Note that each mapping is constrained by the former mapping choices for vertices along the line, and by the height of the interval covered by the selected edge. We define the best mapping as the one that minimizes the total length of the resulting 3D curve (see Figure 5 as a result). This length is computed by splitting the edges in the second view according to the associated vertices from the first view, and by using 3D coordinates of all vertices (the x coordinate of the vertices in the side view being inferred by interpolating those of their surrounding vertices from the front view). We solve this constrained mapping problem using a standard dynamic programming algorithm [Cormen et al. 2001].

The vertices of the reconstructed, 3D feature curves are then projected back to the visual hull for feature-aware meshing. Results are depicted in Figure 5.

The hull is tessellated using a 2D constrained Delaunay triangulation of the projection of the hull faces onto the viewing plane, where all feature curves are used as constraints. The tessellated hull therefore includes all the necessary edges for exactly matching the sketches (Figure 3(c)).

The tessellated hull is finally smoothed out into a *draft model* while maintaining feature matching with the sketches: a vertex of a feature curve drawn on a single sketch is allowed to slide along a straight line, else it is assigned to its reconstructed 3D position. In our implementation, we use the simple smoothing method from [Taubin 1995] and we do not preserve silhouettes while smoothing, since they will be restored in the optimization loop. See Figure 3(d).

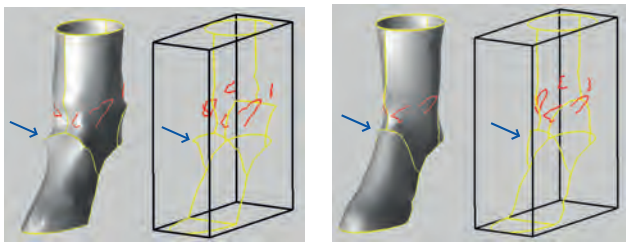

Fig. 5. Without (left) and with (right) 3D reconstruction of seams based on minimal length. Left: When a seam line is visible in 2 sketched views, as it is the case here, a simple concatenation leads to an elongated angular curve on the model (see curve pointed by arrow). Right: The 3D reconstruction in the initializtion step reduces the curve length while still matching with the sketches.

Optimization loop and challenges: Starting from the draft model, step 2 of the processing pipeline optimizes developability while ensuring that the model exactly fits all silhouettes in the multi-view sketches. It therefore alternates developability optimization with silhouette constraints enforcement. As in previous work, silhouette matching makes the model locally inflate or deflate. However two problems remain to be solved:

—Since we make neither symmetry nor flat silhouette assumptions ensuring that silhouettes edges are fully covered and never hidden by other parts of the model is rather intricate. Our solution is described in Section 4.
—Moreover, in addition to 2D and 3D positional constraints due to border and seam lines, pre-designed folds have to be modeled and maintained throughout the process. Our solution is presented in Section 5.

Convergence: We need a global developability measure to decide when to exit step 2. Let the weighted angular defect $D(V)$ be

the absolute value of the sum of face angles around the vertex V, minus 360 degrees, and divided by the area of the 1-ring around V. Our global measure is the percentage of mesh vertices for which $D(V) < 5^\circ/m^2$. In addition to being a local optimum for this measure, the mesh we return meets all the other constraints. We also return the 2D patterns computed by unfolding surface panels using ABF++ [Sheffer et al. 2005] and a color map of the weighted angular defects at each vertex.

Since the sliding constraints method we are using interleaves developability optimization and silhouette matching steps, convergence can only be reached if the silhouette matching does not spoil the current level of developability. We addressed this issue by developing a silhouette matching method (Sect. 4) that best preserves all normals. Then, if the Gauss map of the surface is close to being 1D (i.e. the surface is close to being developable), it tends to remain so. In practice, convergence was reached in less than 10 iterations for all our examples.

4. NON-PLANAR SILHOUETTE MATCHING

We are seeking an efficient method to be used at each iteration of step 2, to ensure that silhouette strokes are fully covered by projections of model points and never hidden by other parts of the model. A simple idea would be to select for each silhouette the set of mesh vertices that protrude the most from the associated viewing direction, and pull them to the closest silhouette point. However, when it is used in a general case (non-planar silhouettes and no local symmetry assumption guaranteeing that silhouette points form a continuous curve), this method tends to select isolated mesh vertices. Pulling the latter typically creates spiky shapes and fails guaranteeing a full coverage of the silhouette, see Figure 6-left.

Standard silhouette detection algorithms will select a subset of edges, called silhouette edges, for which one adjacent polygon is front-facing and one is back-facing [Markosian et al. 1997]. Unfortunately, one observes [Corrêa et al. 1998; Hertzmann and Zorin 2000] that in mesh regions similar to cusps or where many faces are nearly edge-on, several silhouette edges can accumulate and thus produce a nasty pattern with intersections when projected onto the viewing plane. Furthermore, a silhouette edge computed using this definition it not necessarily the most protruding edge and some of them may not be visible from the view point of interest. Pulling these edges onto the silhouette stroke would therefore lead to too large mesh deformations.

Hertzmann and Zorin [2000] approximate smooth surface silhouettes defined as the zero-set of the non-linear function $f(p) =< n(p), p - c >$, where n is the surface normal and c the view point. By assuming that f varies linearly along the mesh edges, the on-edge silhouette points are computed using linear interpolation of f and then connected together. The fact, that the edges of the computed silhouette curve do not belong to mesh edges makes them not suited to be integrated as hard constraints into our developability preserving fitting procedure (as described in Section 4.2).

Silhouette matching combined with Laplacian mesh editing as proposed by Nealen et al. [2005] also does not fit our needs for the following two reasons. Firstly, silhouette matching is herein only a design handle and not a hard constraint. User-prescribed target silhouettes are expressed as soft constraints in a least-squares fitting process, so they are not exactly matched. This is fine for this application, where the detail-preserving nature of Laplacian editing should not be lost. In contrast, our method aims at having silhouette strokes exactly matched. Secondly, the silhouette edges on our model may be arbitrarily segmented (see Figure 8) in which case the method by Nealen et al. is not applicable. Handling segmented model silhouettes is a desired property in our case, since it provides our iterative loop with the necessary flexibility to converge to a developable surface with predefined folds. We would not be able to converge to the boots in Figure 1(b) using continuous silhouette curves on the model as assumed in previous work.

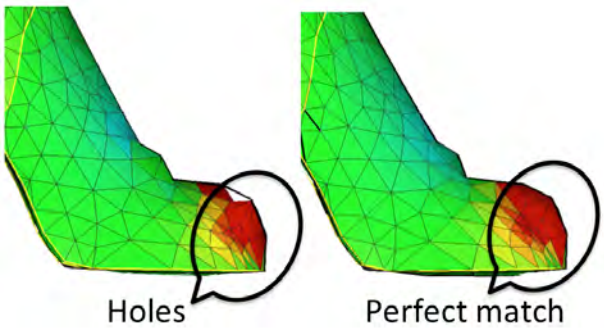

Fig. 6. Constraining isolated vertices to match the silhouette may lead to incomplete silhouettes (left) while our edge based approach ensures perfect silhouette match.

We now develop a novel solution to select a subset of mesh edges based on the three following requirements: Firstly, our model being a triangle mesh, the sets of points on a silhouette can be represented, without loss of generality, as a subset of the mesh edges (a triangle will be entirely on the silhouette if two of its edges are tagged as silhouette edges). Therefore we select a set of edges that we call *s-edges*, to be tied to silhouette strokes rather than mesh vertices, see Figure 6.

Secondly, to ensure that a silhouette stroke from the sketch is entirely covered and well sampled by the projection of s-edges, we exploit the way the draft mesh was built: For each segment of a 2D silhouette stroke, we select s-edges among those built from this specific segment when the tessellated hull was generated. Moreover, each selected edge is assigned to its original projection on the silhouette, leading to an exact reconstruction of the silhouette stroke and ensuring that model points optimally sample the silhouette.

Lastly, silhouettes should not be hidden by other parts of the model. To achieve this, in addition to selecting the edges that protrude most from the relevant viewing direction, we generate a mesh deformation that best preserves normals. The selected edges then tend to remain the most protruding ones after deformation.

4.1 Selecting silhouette edges through zippering

For each silhouette stroke, we use a new **zippering algorithm** to select s-edges such that all the segments of the stroke are covered: the method progressively selects the most protruding edges of the current model while it marches along the silhouette stroke.

Without loss of generality, let the silhouette stroke lie in the (x, z)-plane, let the y-axis be the relevant viewing direction, and let S be a segment of the stroke, see Figure 7(a). As stated in Section 3.3, S was used to generate a planar face on the visual hull. This hull was then triangulated according to seam and fold lines. Let $H(S)$ be the triangulation of the planar hull panel corresponding to segment S (Figure 7(b)) and $M(S)$ be the current corresponding mesh panel before silhouette marching. Since the planar hull triangulation $H(S)$ is in 1-to-1 correspondence with the deformed mesh panel $M(S)$, one can consider $H(S)$ being the pa-

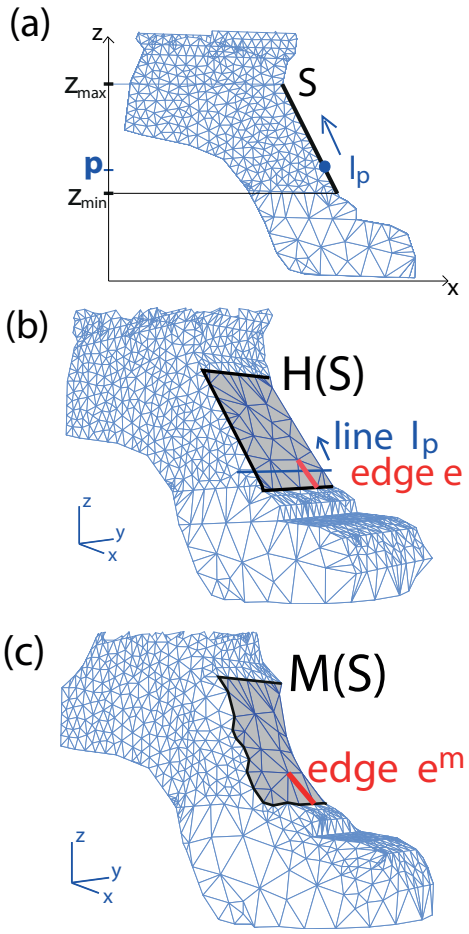

Fig. 7. Protruding edges composing the silhouette: (a) Silhouette with selected silhouette line S and marching parameter p. (b) Silhouette hull triangulation H with marching line l_p. As the marching line advances, we select the most protruding edge among those intersecting l_p. (c) Current mesh M with selected edge in red.

rameter domain of $M(S)$, where each edge $e \in H(S)$ uniquely corresponds to an edge $e^m \in M(S)$.

For each segment S the zippering edge-selection algorithm works in either the x- or z-direction, depending on in which direction the segment spans the most. Assume that the z-direction is selected as in Figure 7(a) and that S spans the interval $[z_{\min}, z_{\max}]$. The algorithm initiates a marching line l_p parallel to the y-axis by setting the marching parameter $p := z_{\min}$. p corresponds to the starting point of silhouette segment S and defines a rim line on the silhouette hull, Figure 7(b). As p is marching from $z_{\min}$ to $z_{\max}$, the marching line l_p slides along the hull $H(S)$. For each specified parameter p, we select, among all edges in $H(S)$ crossing l_p, the most protruding edge e as follows:

$$\bar{e} = \max_{\substack{e \in H(S) \\ p \in [z_1(e), z_2(e)[}} \Big(\text{sign}(e^m) \cdot dist(e, e^m) \Big), \qquad (1)$$

where $z_1, z_2 \in [z_{\min}, z_{\max}]$ with $z_1 \leq z_2$ are z-coordinates of vertices of e, $dist(e, e^m)$ is the distance between the edge midpoints and $\text{sign}(e^m)$ is positive in case the edge e^m is lying outside the silhouette hull, negative otherwise. p is then updated to $p = z_2(\bar{e})$, and the algorithm iterates the edge selection (1) until $p \geq z_{\max}$.

The output of the zippering algorithm is a list of edges $\{e\}$ on the tessellated hull and their counterparts $\{e^m\}$ on the actual mesh. The edges of the hull exhibit small overlaps in the (x, z)-plane, but have different y-depths. These overlaps will guarantee a complete covering of the silhouette. It may happen that during edge selection short edges are "hidden" by longer edges. This is however not a problem in practice: indeed, if the non-selected shorter edge was still protruding after this step, it would be selected at the next iteration. Note that if the two edges are not too far away, the selection of the larger edge followed by the developability optimization step may already solve the problem, since the displacement of the selected edge propagates in a neighborhood.

Note that the method seamlessly handles edge selection when the model is inside, versus outside the silhouette. Figure 8 shows the set of s-edges $\{e^m\}$ on the mesh at different steps of the optimization loop.

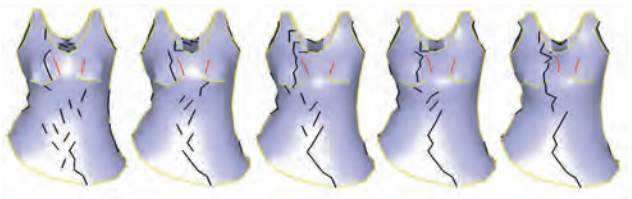

Fig. 8. Sliding constraints: during optimization (step 2), the set of edges tied to the side-view silhouette (black) progressively slides over the surface, depending on its deformation, so that the silhouette stroke is always exactly reconstructed by the edges that protrude the most.

4.2 Silhouette matching

Once the set $\{e^m\}$ of the most protuding mesh edges has been selected, we deform the mesh in order to match the silhouette. The idea is to enforce silhouette constraints by moving the edges $\{e^m\}$ to adapted target positions while maintaining all 2D and 3D positional constraints and best preserving all triangle normals, see Figure 9 for a result. This is done by adapting the As-2D-as-possible deformation method of [Brouet et al. 2012], as follows:

As-2D-as possible deformation: This method sets mesh triangles to target orientations in a single linear step. Triangles are allowed to change shape and size to maintain connectivity. Let $(\boldsymbol{p}_1, \boldsymbol{p}_2, \boldsymbol{p}_3)$ be the vertices of triangle t and $\boldsymbol{p}_4$ defined by offsetting $\boldsymbol{p}_1$ in the normal direction. The 3×3-matrix $P^t = (\boldsymbol{p}_4 - \boldsymbol{p}_1, \boldsymbol{p}_4 - \boldsymbol{p}_2, \boldsymbol{p}_4 - \boldsymbol{p}_3)$ representing the local 3D frame of t is used to define the deformation transfer from the source triangle to the unknown target triangle $(\tilde{\boldsymbol{p}}_1, \tilde{\boldsymbol{p}}_2, \tilde{\boldsymbol{p}}_3)$ as $\tilde{P}^t (P^t)^{-1}$. $\tilde{P}^t$ is the frame of t after deformation. The deformation is expressed as the least-squares minimization of:

$$\sum_t \| \tilde{P}^t (P^t)^{-1} - T^t \|_F^2, \qquad (2)$$

where $\|M\|_F = (\sum_i \sum_j |m_{ij}|^2)^{1/2}$ is the Frobenius matrix norm and T^t the known gradient which rotates triangle t into a target orientation. In our case, we want to preserve orientation, so T^t is set to the identity matrix. Two types of constraints are then added when minimizing (2).

Feature line constraints: We take benefits of the feature aware tessellation used in step 1 of the processing pipeline where a set of mesh edges is pre-associated with feature lines (seam or fold). When a feature was drawn in two views, it results in a set of 3D positional constraints. So, the corresponding x, y and z-coordinates are set to these position values and are removed from the set of unknowns of equation (2). When it was drawn in a single sketch (e.g. the (x, z)-plane), the sketched feature imposes a set of 2D constraints. Again, we simply set fixed values for the coordinates specified on the sketch for these vertices, remove these coordinates from the unknowns of equation (2) and only allow the depth of the vertices to change.

Silhouette constraints: Let $\boldsymbol{p}_1, \boldsymbol{p}_2$ be the vertices of a selected s-edge $\{e^m\}$ and $\boldsymbol{v}_1, \boldsymbol{v}_2$ be vertices of its counterpart $\{e\}$ on the tessellated hull $H(S)$. To ensure an exact reconstruction of the silhouette stroke, we set the x and z coordinates of $\{e^m\}$ back to those of e: $\bar{p}_i^x = v_i^x$ and $\bar{p}_i^z = v_i^z$ and remove the two pre-set coordinates per vertex of each s-edge from the unknowns before minimizing (2).

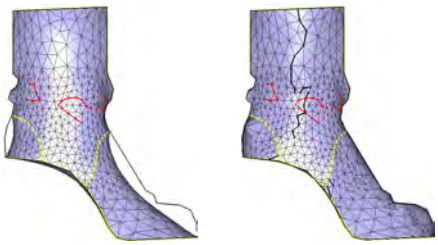

Fig. 9. Silhouette matching: Projection of most protruding edges to the silhouette and mesh deformation best preserving normals. The silhouettes strokes of the side-view are perfectly matched, as well as the 3D feature line constraints (yellow). The mesh edges marked in black on the bootleg are the selected edges tied to the silhouette stroke of the front view.

Least square minimization: Following the same steps than Sumner and Popovic [2004], the minimum of expression (2) can be found by solving a linear system with respect to the unknowns $\tilde{\boldsymbol{p}}_i$. The best solution is computed as a standard least square minimization using the normal equations of the associated linear system. The resulting $(\tilde{\boldsymbol{p}}_1, \tilde{\boldsymbol{p}}_2, \tilde{\boldsymbol{p}}_3)$ vertices define the new geometry of the mesh. As the linear system is separable in the (x, y, z)-coordinates, we actually solve three smaller systems in each coordinate. Therefore, for a mesh of V vertices, T triangles and c hard constraints for a given coordinate, we solve three linear systems of size $V + T - c$. Note. that c may be different for each coordinate. The solution can be computed efficiently since our meshes are quite coarse. Note that the system could be further reduced using the formulation in [Botsch and Sorkine 2008].

5. HANDLING INTERNAL, PRE-DESIGNED FOLDS

The method we have described so far builds a surface that exactly matches the silhouettes and seam-lines in multi-view sketches. The last challenge we need to solve is to handle the pre-designed folds also depicted on the input sketches and to optimize developability without flattening these folds. Our solutions for representing and maintaining folds are discussed below. In particular, we propose a new, fold-preserving method for optimizing developability.

5.1 Folds from sketched contours

A first idea could be to model folds as internal silhouettes, enabling us to re-use the matching process we just described. However, looking at many design sketches such as those in Figures 1 and 2, we noted that most of sketched folds are not so extreme. They rather correspond to a region where the surface is either bumping inwards or outwards. Moreover, they are generally represented by their contours rather than by their center-line. Based on this observation, we follow the idea from [Popa et al. 2009] to use an orientation constraint for modeling the folds, but we propose a simpler geodesic diffusion approach and actually generate the folds by fitting prescribed target orientations for the mesh triangles. In contrast with [Popa et al. 2009], our formulation avoids solving several linear systems over the whole mesh. It is thus much more efficient and still sufficient for generating plausible initial fold shapes. The orientation constraints we use consist in restricting the normal vectors of the mesh triangles along a fold contour to form a fixed angle with the viewing direction. [Popa et al. 2009] found out empirically that the range of 30°-60° best characterizes folds from wide to narrow size. We use 60° in our implementation in order to get well pronounced folds.

To get a better starting point for optimization, we take folds into account at the end of step 1 of the processing pipe-line (see Section 3.3). Then, they are maintained and their shape is made more developable within the optimization loop of step 2.

5.2 Initializing folds

An offset distance λ is defined to control the *region of influence* (ROI) of the deformation due to each fold line. In practice, λ is automatically set from material thickness value preselected by the user, as was done in [Rohmer et al. 2010]. Fold contours are reconstructed by the tessellated mesh as are the other feature lines (see red curves in Figure 3(c)). To ensure that the normal vectors along them exhibit the pre-set rotation angle, the triangles along the curve are then rotated around the edge that belongs to the fold contour. To preserve surface smoothness, we first diffuse the orientation constraints along fold contours.

The rotation induced by a fold-curve segment s on a mesh triangle t is pre-set to the rotation around s that brings a 60 degrees angle with the viewing direction. The magnitude of this rotation is then scaled, depending on the geodesic distance from s to t, by a weight $\omega_t^s \in [0, 1]$ that falls down to zero at the border of the ROI. Let $\boldsymbol{n}_t^s$ be the triangle normal after this rotation. If t is within the ROI of several fold segments (belonging to the same or to different fold lines), we compute its target normal $\boldsymbol{n}_t$ using quaternion interpolation of $\boldsymbol{n}_t^s$. Finally, mesh triangles are re-oriented to the resulting target orientations while maintaining mesh connectivity (see Figure 3(d)), using the As-2D-as-possible deformation formulation of Equation (2) with the gradient matrix T^t set to the rotation matrix with axis and angle given resp. by $(\boldsymbol{p}_4 - \boldsymbol{p}_1) \times \boldsymbol{n}_t$ and $\arccos((\boldsymbol{p}_4 - \boldsymbol{p}_1) \cdot \boldsymbol{n}_t)$. Preserving 2D and 3D positional constraints while minimizing (2) just requires removing a few unknowns, as explained in Section 4.

5.3 Folds-preserving developability optimization

The silhouette matching step of the optimization loop will naturally preserve folds, since it best preserves all normals. The same requirement of folds preservation holds for the developability optimization step described in this section. However, the method has to be chosen with care: using standard local surface flattening methods would indeed improve developability, but at the price of loosing

folds. Further requirements for this step are the efficiency (since it will be used within an iterative process) and the ability to preserve positional constraints. We therefore developed an appropriate method, described next.

Since we are dealing with surfaces possessing folds, which locally behave like cones or cylinders, we propose to improve developability while preserving the existing folds by locally fitting the best approximating cones. We found the most appropriate developability measure in [Julius et al. 2005; Decaudin et al. 2006]. It is a metric that builds on a common property of cones, cylinders and planar patches that is: the surface normal vector $\boldsymbol{n}$ has a constant angle Θ with an axis $\boldsymbol{A}$. The local fitting error derived from this measure for a mesh triangle t is thus:

$$(\boldsymbol{n}_t \cdot \boldsymbol{A} - \cos\Theta)^2. \quad (3)$$

[Decaudin et al. 2006] propose a complex three-step algorithm to minimize this error. The three steps are a non-linear optimization process to find the best approximating cone, followed by two distinct fitting steps. Based on the developabiity error measure (Equation (3)), we propose a different algorithm which meets all our requirements but uses only two linear steps. It improves developability, preserves the existing folds and simultaneously satisfies all the extra positional and normal constraints.

The first step of our method is to compute the best fitting cone (defined by A and Θ) to each triangle's neighborhood in order to obtain its target normal vector. Let us define Θ by:

$$\cos\Theta := \hat{\boldsymbol{n}} \cdot \boldsymbol{A}, \quad (4)$$

where $\hat{\boldsymbol{n}}$ is the average normal vector in the neighborhood $N(t)$ of a triangle t. We first estimate the axis $\boldsymbol{A}$ by solving

$$\arg\min_{\boldsymbol{A}} \sum_{j \in N(t)} ((\boldsymbol{n}_j - \hat{\boldsymbol{n}}) \cdot \boldsymbol{A})^2 \quad \text{subject to} \quad \|\boldsymbol{A}\| = 1, \quad (5)$$

where the error term $((\boldsymbol{n}_j - \hat{\boldsymbol{n}}) \cdot \boldsymbol{A})$ is obtained by replacing (4) into (3). Problem (5) is known as a total least squares minimization. The solution $\boldsymbol{A}$ is the vector corresponding to the smallest singular value (in absolute value) of the matrix composed of the vectors $\boldsymbol{n}_j - \hat{\boldsymbol{n}}$, the constraint $\|\boldsymbol{A}\| = 1$ is automatically satisfied. We then compute Θ from (4). In our implementation we choose N as the 1-neighborhood of a triangle's edges and vertices. Even though a larger N may improve the approximation, in our setting the quality of the approximation will however be improved anyway when this step is repeated in the whole optimization loop. The target normal vector $\boldsymbol{n}$ for each mesh triangle is computed by rotating the actual triangle normal $\boldsymbol{n}_t$ around the vector $\boldsymbol{n}_t \times \boldsymbol{A}$ such that $(\boldsymbol{n} \cdot \boldsymbol{A} - \cos\Theta = 0)$ after rotation. However, we leave the target normal unchanged for the triangles adjacent to a fold line, since it was already pre-set to account for the fold orientation with respect to the viewing direction.

In a second step, the mesh deformation best approximating all target normals is computed using our extension (Sect. 5.2) of the As-2D-as-possible approach (2), enabling us to take the 2D and 3D positioning of features on the sketches into account as well. A further benefit of this new approach for optimizing developability lies in its efficiency, since only a single linear system needs to be solved.

Note that this method alone is not able to remove isolated singularities such as the tip of a cone: Being based on convergence to the closest local conical approximation, it would leave such local conical features unchanged. To favor smooth developable surfaces, we add a local developability pass at the end of this process. If the angular defect around a vertex is larger than a threshold, we apply one step of the local flattening operation in [Wang and Tang 2004]: The vertex is re-positioned in the normal direction to minimize the local angular defect.

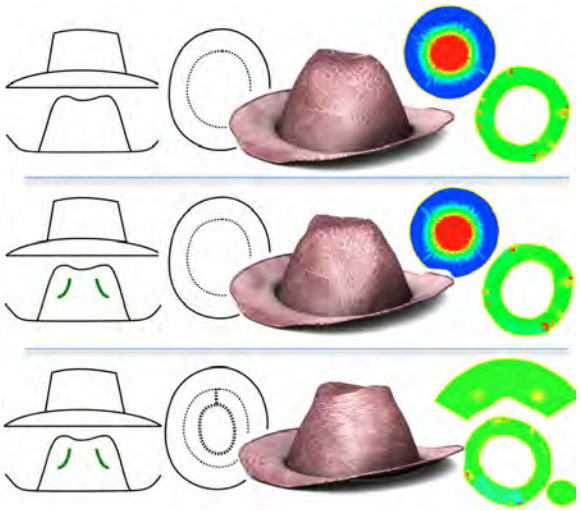

Fig. 10. Progressive design: sequence of sketches and resulting models with patterns. Top: Initial sketch. Middle: Folds-lines added. The error-map on the patterns still indicates a result being far from being developable. Bottom: Seams are added on the top view to improve developablity. Note, that in the first two cases the pattern is composed of 2 parts, one of which has a high developability error. On the bottom example, seam lines were added on the top view, resulting into 3 developable parts.

6. RESULTS AND DISCUSSION

Sketching interface: In our implementation, the user selects a different pen to draw either seam and border lines, external silhouettes or fold contours (yellow, black and red lines respectively shown in Figure 3(a) and Figures 11-14-left). Pre-existing fashion sketches can be displayed as a background image and used as a guide during this input step. A fold being represented by contour lines, its inward or outward nature is set from the sketching direction and can then be switched manually. When a feature line (seam, border or fold) is sketched in two views, our interface displays guidelines and provides snapping mechanisms to assist the designer in defining a coherent second view. The user can stop sketching at any point and run the method: if the user starts from coarse sketches with missing seam lines, the current surface panels will be far from developable, but the 3D model will already give a good idea of the current model. The error map representing local angular defects (see Section 3.3) enables the user to progressively improve the design, as shown in Figure 10.

Examples: Figures 1 and 10-17 show a number of results from our method. Several of them where inspired from pre-existing design sketches such as those of Figure 2, enabling us to validate the use of our method for arbitrary models. The examples in Figures 10 and 14 were actually created by a cloth designer. In all our examples the emphasis is on the generation of non-planar silhouettes and predesigned folds while producing developable surfaces.

The phrygian cap shown in Figure 13 validates our sliding constraints paradigm in an extreme case. This particular highly non-planar silhouette is produced progressively by our system in con-

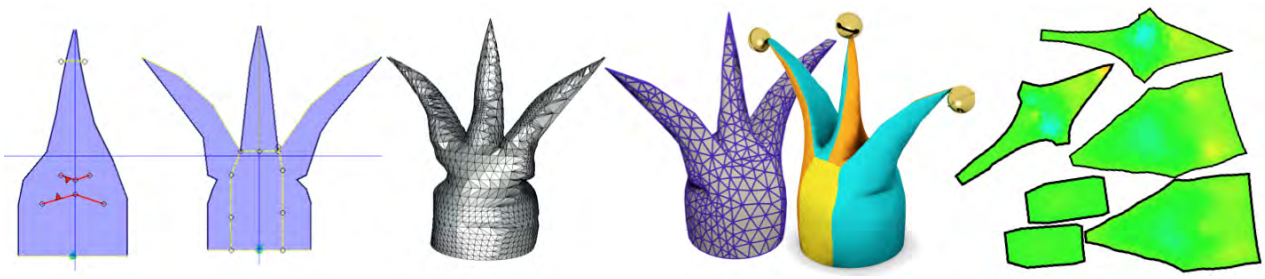

Fig. 11. Joker hat sketched with our system and comparison with [Rivers et al. 2010]'s in the middle (gray mesh).

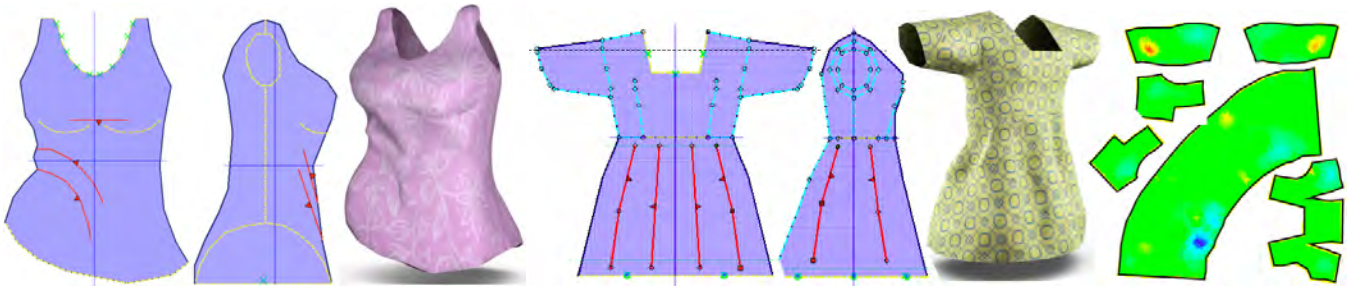

Fig. 12. Shirt and Tunic sketched with our system.

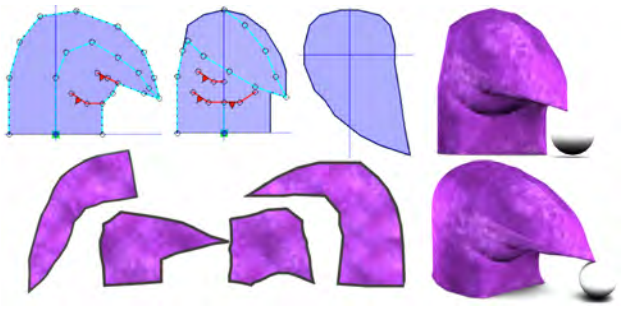

Fig. 13. Highly non-planar silhouettes are used to design this Phrygian cap.

cordance to the three input sketches without any user-interaction. The phygian cap and the hat (Figure 10) are the only models for which three sketches, including a top view, were necessary to fully specify the shape.

Concerning the sliding silhouette paradigm, one can observe in practice that silhouette curves tend to converge from sets of disjoint edges to several pieces of contiguous silhouette curves. This is due to the smoothing effect of developability optimization steps: We converge to a surface that meets all constraints, embeds folds and is nearly developable. Therefore, the initially rough surface becomes smoother and smoother, leading to a more continuous set of silhouette s-edges. This is a pleasant side effect that we do not control explicitly.

For the boot, we show both the designer's sketch (Figure 1(a)) and the input of our system (Figure 3(a)). The boot illustrates a leather object, for which folds are typically part of design. The color-coded 2D patterns, computed with [Sheffer et al. 2005], validate the good level of developability of the surface we generated, except at the tip of the boot (red region on the pattern). We further validated our approach by sewing an actual leather boot from the patterns (Figure 1(d)).

The joker hat (Figure 11) validates our method for sketches with complex, curved silhouettes. It fits well to the sketches and brings a very different solution from the one computed from the multi-view sketching method in [Rivers et al. 2010], shown for comparison: While the grey mesh in the middle-left (Rivers et al's method) does match the input sketches its lower part is a generalized cylinder of varying thickness (which makes it look somewhat as a chess piece). The deep concavities at the top of this lower part are regions with non-zero Gaussian curvature, since curvatures in the vertical and horizontal directions are both obviously non-zero. This makes the surface non-developable. This is corrected by our solution shown at the right. We even achieved a very good level of developability in this case, as shown by the error map displayed on the patterns, even though surface panels have non-trivial geometry. Note also the smoother and flatter shapes we get for the three upper parts.

The shirt, the tunic and the coat (Figure 12, 14) give further examples of mannequin-free garment design. Note that we set no symmetry constraint on the patterns of these models, leaving our designers free to sketch whatever they imagined. All examples converged in 3 to 7 iterations. See table 1 for full statistics. The leather coat is shown in another posture, computed using physically-based simulation under collision and gravity forces, in Figure 17. We also included a full animation of the tunic in the associated video, to validate the fact that the models we output are ready for animation. Note however that the meshes we output should be refined before simulation if thinned cloth material was to be modeled.

Finally, we would like to stress that our "sketching folds" method is particularly well suited (although not exclusively) to design of products with pre-designed folds. Our last two examples of a doctor bag and a smaller purse in Figure 15 obviously belong to this class of products. All the folds drawn by the designer in the front and side view sketches are intended to remain permanently. They

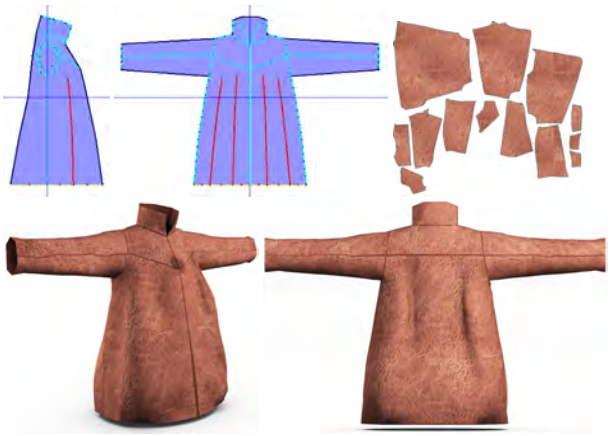

Fig. 14. ShortCoat designed with our system.

clearly appear faithful to the sketch in the resulting 3D model. Note that silhouette curves are totally non-planar for these two models (as highlighted by yellow curves in Figure 16), preventing them to be created using former sketch-based modeling systems.

Limitations of the method: Although being able to build quasi-developable surfaces matching multi-view sketches is a useful achievement, we noticed several limitations of our method.

Firstly, there may be a lack of user-control when complex shapes are reconstructed from only a couple of sketches. The joker hat (Figure 11) gives a good illustration of the problem. Although the top of the hat has three parts, our designer was only enabled to sketch a single side-view silhouette: at each step of the iterative optimization loop, our method ensures that one part of the hat matches each point of this silhouette. The other parts are only constrained to be somewhere inside. Enabling the user to divide the shape into several parts, and sketch partial silhouettes for each of them, would be a useful extension when more control is desired. Another part of our method where control may be lacking concerns pre-designed folds: Our orientation-based method enabled us to model simple folds, which are then improved thanks to developability optimization. However, providing a better control of fold shapes at the initialization stage would be useful. Using the implicit model for geometrical folds in [Rohmer et al. 2010] could be a good choice, since it allows to model merging folds as well as isolated ones.

Secondly, the shape we design can currently be set to be symmetric by using symmetric input sketches. However, in the case of garments, being able to enforce symmetry on the pattern rather than on the 3D model could be even more useful. This is still an open problem.

Lastly, the generated surface is not guaranteed being free of self-intersection. In practice, we only obtained a self-intersection in one of our examples, namely a former version of the shirt of Figure 12. Removing such intersection currently needs to be done as a post-process using a collision processing method.

7. CONCLUSION

We have presented the first method enabling to generate smooth developable surfaces with folds, such as cloth or leather products, from multi-view design sketches. The main contribution is the fact we seamlessly handle non-planar silhouettes. In particular, we use sliding constraints to progressively match silhouettes while optimizing developability. We introduce a zippering algorithm enabling selected s-edges to exactly match silhouette strokes. And we pro-

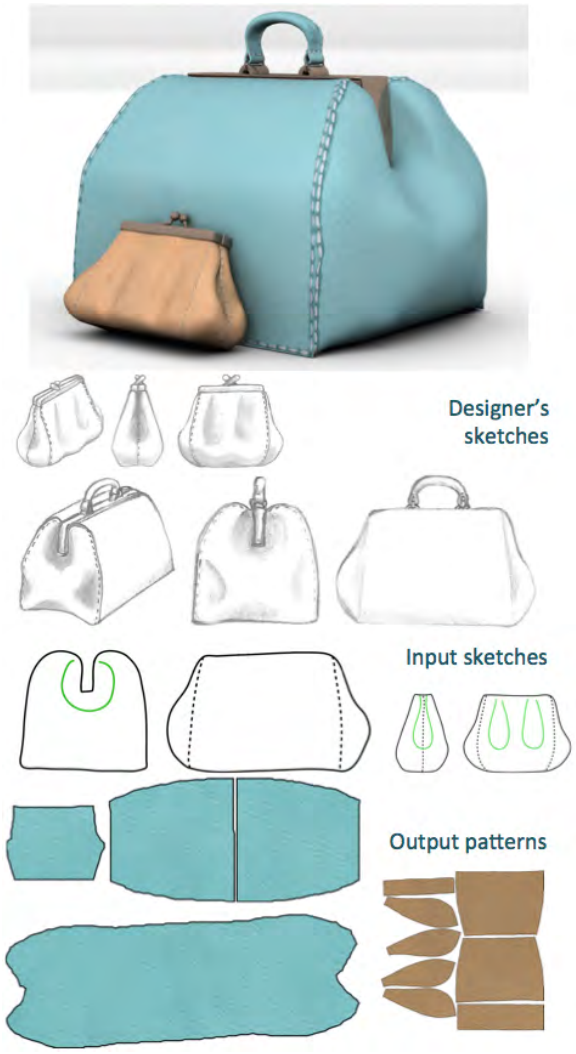

Fig. 15. Top: Doctor bag and leather purse designed with our system. Middle: designer's sketches exhibiting design folds which are indended to remain permanently on the product and the sketches given as input to our system. Bottom: the patterns we output. Note that we did not use symmetry for these examples: therefore, neither the input sketches nor the resulting shapes or patterns are fully symmetric.

Model	#V	t_1	% it	t_2	% dev. final	L_2 stretch	L_∞ stretch
boot	1665	4.1	3	3.4	94.5	1.022	1.876
Joker	2297	2.1	7	4.2	96.8	1.023	1.223
Shirt	1978	3.9	5	4.5	95.9	1.021	1.526
Tunic	1811	3.2	5	4.6	95.8	1.010	3.203
Phrygian	1388	1.8	5	3.4	97.2	1.015	1.326
Hat	1904	2.6	4	3.7	95.3	1.010	1.204
Coat	2338	4.9	6	4.8	95.6	1.040	2.204

Table 1: Result statistics. Number of vertices #V, preprocessing time in seconds t_1, number of iterations (step 2) #it, time per iteration in seconds t_2, percentage of vertices with angular defect less than $5^\circ/m^2$, L_2 and L_∞-stretch.

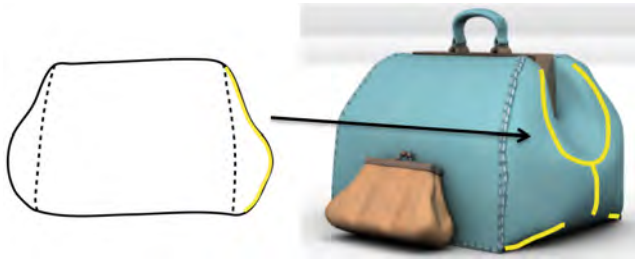

Fig. 16. Left: Input sketch with one silhouette highlighted in yellow. Right: Doctor bag with the corresponding non-planar silhouettes curves highlighted in yellow.

pose an adaptation of the As-2D-as-possible method for optimizing developability while preserving pre-designed folds.

Designing developable models from a set of two or three axial views is a current practice for designers of leather or cloth products. Our method, which has already been validated and used by a few of them, is therefore a promising technique for industrial applications: combined with recent interactive post-editing techniques [Umetani et al. 2011], it would enable designers to quickly convert a conceptual idea into a first 3D model, later fully simulated and improved.

In terms of sketch-based modeling challenges in Computer Graphics, a very interesting related problem, to study next, would be the generation of developable models from a single, perspective sketch, such as the hat in Figure 2. The fact that we are able to imagine a 3D shape from this type of sketch can give some hope. Such perspective sketches, possibly drawn with a photo as background image and with no correspondence to maintain between several views, could be an easier 3D modeling interface for beginners.

Fig. 17. Mannequin in a different posture, dressed with the hat (Figure 10), the boots (Figure 1) and coat (Figure 14) designed with our system. This simulation frame (shown in 4 different view) was computed using a commercial physically-based simulation package. Here pre-designed folds for the coat (corresponding to more leather material and to larger 2D patterns) seamlessly combine with those due to this specific posture under gravity.

ACKNOWLEDGMENTS

This research was partially funded by the ERC Advanced Grant Expressive (ADG-2011 291184) and the Equipex ANR-11-EQPX-0002.

REFERENCES

BAE, S.-H., BALAKRISHNAN, R., AND SINGH, K. 2008. Ilovesketch: As-natural-as-possible sketching system for creating 3d curve models. In *Proceedings of the 21st Annual ACM Symposium on User Interface Software and Technology*. UIST '08. ACM, New York, NY, USA, 151–160.

BO, P. AND WANG, W. 2007. Geodesic-Controlled Developable Surfaces for Modeling Paper Bending. *Computer Graphics Forum 26,* 3, 365–374.

BOTSCH, M. AND SORKINE, O. 2008. On linear variational surface deformation methods. *IEEE Transactions on Visualization and Computer Graphics 14,* 1, 213–230.

BROUET, R., SHEFFER, A., BOISSIEUX, L., AND CANI, M.-P. 2012. Design preserving garment transfer. *ACM Transactions on Graphics (TOG), Proceedings of ACM SIGGRAPH 31,* 4, 36:1–36:11.

CLO3D. Virtual fashion inc. (http://www.clo3d.com/).

CORMEN, T. H., LEISERSON, C. E., RIVEST, R. L., AND STEIN, C. 2001. *Introduction to Algorithms, Second Edition*. The MIT Press and McGraw-Hill Book Company, Cambridge , Massachusetts London, England.

CORRÊA, W. T., JENSEN, R. J., THAYER, C. E., AND FINKELSTEIN, A. 1998. Texture mapping for cel animation. In *Proceedings of the 25th Annual Conference on Computer Graphics and Interactive Techniques*. SIGGRAPH '98. ACM, New York, NY, USA, 435–446.

CUTLER, L. D., GERSHBEIN, R., WANG, X. C., CURTIS, C., MAIGRET, E., PRASSO, L., AND FARSON, P. 2007. An art-directed wrinkle system for CG character clothing and skin. *Graphical Models 69,* 56, 219 – 230. Special Issue on SCA 2005.

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 (CGF). Proceedings of Eurographics 25,* 3, 625–634.

FREY, W. H. 2004. Modeling buckled developable surfaces by triangulation. *Computer Aided Design (CAD) 36,* 4, 299–313.

HARMON, D., PANOZZO, D., SORKINE, O., AND ZORIN, D. 2011. Interference-aware geometric modeling. *ACM Transaction on Graphics (TOG) 30,* 6, 137:1–137:10.

HERTZMANN, A. AND ZORIN, D. 2000. Illustrating smooth surfaces. In *Proceedings of the 27th Annual Conference on Computer Graphics and Interactive Techniques*. SIGGRAPH '00. ACM Press/Addison-Wesley Publishing Co., New York, NY, USA, 517–526.

JULIUS, D., KRAEVOY, V., AND SHEFFER, A. 2005. D-charts: Quasi-developable mesh segmentation. *Computer Graphics Forum 24,* 3, 581–590.

KILIAN, M., FLÖRY, S., CHEN, Z., MITRA, N. J., SHEFFER, A., AND POTTMANN, H. 2008. Curved Folding. *ACM Transaction on Graphics (TOG), Proceedings of ACM SIGGRAPH 27,* 3, 75:1–75:9.

LIU, Y., POTTMANN, H., WALLNER, J., YANG, Y.-L., AND WANG, W. 2006. Geometric Modeling with Conical Meshes and Developable Surfaces. *ACM Transaction on Graphics (TOG), Proceedings of ACM SIGGRAPH 25,* 3, 681–689.

MARKOSIAN, L., KOWALSKI, M. A., GOLDSTEIN, D., TRYCHIN, S. J., HUGHES, J. F., AND BOURDEV, L. D. 1997. Real-time nonphotorealistic rendering. In *Proceedings of the 24th Annual Conference on Computer Graphics and Interactive Techniques*. SIGGRAPH '97. ACM Press/Addison-Wesley Publishing Co., New York, NY, USA, 415–420.

MARVELOUS DESIGNER. http://www.marvelousdesigner.com/.

MORI, Y. AND IGARASHI, T. 2007. Plushie: an interactive design system for plush toys. *ACM Transaction on Graphics (TOG), Proceedings of ACM SIGGRAPH 26,* 3.

MÜLLER, M. AND CHENTANEZ, N. 2010. Wrinkle meshes. In *Proceedings of the 2010 ACM SIGGRAPH/Eurographics Symposium on Computer Animation*. SCA '10. Eurographics Association, Aire-la-Ville, Switzerland, Switzerland, 85–92.

NEALEN, A., SORKINE, O., ALEXA, M., AND COHEN-OR, D. 2005. A sketch-based interface for detail-preserving mesh editing. *ACM Transaction on Graphics (TOG), Proceedings of ACM SIGGRAPH 24,* 3, 1142–1147.

OPTITEX. Optitex ltd. (http://www.optitex.com/).

PETERNELL, M. 2004. Developable surface fitting to point clouds. *Computer Aided Geometric Design (CAGD) 21,* 8, 785–803.

POPA, T., ZHOU, Q., BRADLEY, D., KRAEVOY, V., FU, H., SHEFFER, A., AND HEIDRICH, W. 2009. Wrinkling Captured Garments Using Space-Time Data-Driven Deformation. *Computer Graphics Forum (CGF) 28,* 2, 427–435.

POTTMANN, H. AND FARIN, G. 1995. Developable rational Bézier and B-spline surfaces. *Computer Aided Geometric Design (CAGD) 12,* 5, 513–531.

PROTOPSALTOU, D., AREVALO, C. L. M., AND MAGNENAT-THALMANN, N. 2002. A body and garment creation method for an Internet based virtual fitting room. In *Proceedings of Computer Graphics International (CGI)*. 105–122.

RIVERS, A., DURAND, F., AND IGARASHI, T. 2010. 3D Modeling with Silhouettes. *ACM Transaction on Graphics(TOG), Proceedings of ACM SIGGRAPH 29,* 4, 109:1–109:8.

ROBSON, C., MAHARIKA, R., SHEFFER, A., AND CARR, N. 2011. Context-aware garment modeling from sketches. *Computers & Graphics 35,* 3, 604 – 613. Shape Modeling International (SMI) Conference 2011.

ROHMER, D., POPA, T., CANI, M.-P., HAHMANN, S., AND SHEFFER, A. 2010. Animation Wrinkling: Augmenting Coarse Cloth Simulations with Realistic-Looking Wrinkles. *ACM Transaction on Graphics (TOG), Proceedings of ACM SIGGRAPH Asia 29,* 5, 157:1–157:8.

ROSE, K., SHEFFER, A., WITHER, J., CANI, M.-P., AND THIBERT, B. 2007. Developable surfaces from arbitrary sketched boundaries. In *Proceedings of the Fifth Eurographics Symposium on Geometry Processing*. SGP '07. Eurographics Association, Aire-la-Ville, Switzerland, Switzerland, 163–172.

SCHMIDT, R., KHAN, A., SINGH, K., AND KURTENBACH, G. 2009. Analytic drawing of 3d scaffolds. *ACM Transaction on Graphics (TOG). Proceedings of ACM SIGGRAPH Asia 28,* 5, 149:1–149:10.

SHEFFER, A., LÉVY, B., MOGILNITSKY, M., AND BOGOMYAKOV, A. 2005. Abf++: fast and robust angle based flattening. *ACM Transaction on Graphics (TOG) 24,* 2, 311–330.

SINGH, K. AND FIUME, E. 1998. Wires: A geometric deformation technique. In *Proceedings of the 25th Annual Conference on Computer Graphics and Interactive Techniques*. SIGGRAPH '98. ACM, New York, NY, USA, 405–414.

SOLOMON, J., VOUGA, E., WARDETZKY, M., AND GRINSPUN, E. 2012. Flexible developable surfaces. *Comp. Graph. Forum 31,* 5, 1567–1576. Symposium on Geometry Processing (SGP).

SUMNER, R. W. AND POPOVIĆ, J. 2004. Deformation transfer for triangle meshes. In *ACM SIGGRAPH 2004 Papers*. SIGGRAPH '04. ACM, New York, NY, USA, 399–405.

TANG, K. AND CHEN, M. 2009. Quasi-Developable Mesh Surface Interpolation via Mesh Deformation. *IEEE Transactions on Visualization and Computer Graphics (TVCG) 15,* 3, 1–1.

TANG, K. AND WANG, C. C. L. 2005. Modeling Developable Folds on a Strip. *Journal of Computing and Information Science in Engineering (JCISE) 5,* 518–528, 35–48.

TAUBIN, G. 1995. Curve and surface smoothing without shrinkage. In *Proceedings of the Fifth International Conference on Computer Vision*. ICCV '95. IEEE Computer Society, Washington, DC, USA, 852–857.

TURQUIN, E., WITHER, J., BOISSIEUX, L., CANI, M.-P., AND HUGHES, J. 2007. A sketch-based interface for clothing virtual characters. *IEEE Computer Graphics & Applications 27*, 72–81.

UMETANI, N., KAUFMAN, D. M., IGARASHI, T., AND GRINSPUN, E. 2011. Sensitive couture for interactive garment modeling and editing. *ACM Transaction on Graphics (TOG), Proceedings of ACM SIGGRAPH 30,* 4, 90:1–90:12.

VOLINO, P., CORDIER, F., AND MAGNENAT-THALMANN, N. 2005. From early virtual garment simulation to interactive fashion design. *Computer-Aided Design (CAD) 37*, 593–608.

WANG, C. C. L. 2008. Towards Flattenable Mesh Surfaces. *Computer Aided Design (CAD) 40,* 1, 109–122.

WANG, C. C. AND TANG, K. 2004. Achieving developability of a polygonal surface by minimum deformation: a study of global and local optimization approaches. *The Vis. Comp. 20*, 521–539.

WANG, H., HECHT, F., RAMAMOORTHI, R., AND O'BRIEN, J. F. 2010. Example-Based Wrinkle Synthesis for Clothing Animation. *ACM Transaction on Graphics (TOG) 29,* 4, 107:1–107:8.

ZHU, L., IGARASHI, T., AND MITANI, J. 2013. Soft folding. *Computer Graphics Forum 32,* 7, 167–176.