Context-Aware Garment Modeling from Sketches

Cody Robson[a,c], Ron Maharik[a], Alla Sheffer[a], Nathan Carr[b]

[a] *University of British Columbia*
[b] *Adobe Systems Inc.*
[c] *NVIDIA Corporation*

Abstract

Modeling of realistic garments is essential for creating believable virtual environments. Sketch-based modeling of garments presents an appealing, easy to use alternative to the established modeling approaches which are time consuming and require significant tailoring expertise. Unfortunately, the results created using existing sketch-based methods lack realism. Driven by human perception of garment sketches, we propose a context-aware garment sketch interpretation based on a set of observations about key factors that affect the shape of garments. Based on this analysis we develop a geometric method for sketch based modeling of garments which obtains more realistic results than previous techniques. We demonstrate the effectiveness of our method on a variety of inputs and validate our approach via a user study where viewers were asked to compare the believability of our outputs versus previous ones.

Categories and Subject Descriptors (according to ACM CCS): Computer Graphics [I.3.3]: Picture/Image Generation—Line and Curve Generation

Keywords: garment modeling; surface modeling; sketch-based modeling

1. Introduction

Modeling of believable virtual garments is essential for creating realistic virtual environments and can benefit other application areas such as clothing design [15]. The traditional approach for modeling virtual garments largely follows the real-life design and tailoring workflow [10, 5, 15]. While it enables the creation of realistic, sophisticated garments, it requires both significant time investment and a high degree of tailoring expertise. Scanning of real life garments, e.g. [1] provides an alternative garment creation approach, but may require even more user time as well as specialized knowledge. With the increasing popularity of avatar-based game environments and on-line design tools it is likely more and more users, who lack such expertise, would want to design virtual outfits themselves.

A user-friendly sketch-based garment modeling approach is proposed by Turquin et al. [24] and built-upon by [3, 21, 25]. With fashion drawings widely used to convey garment shapes, sketching provides a particularly natural modeling interface, well-suited even for users who are not computer savvy. Unfortunately, some of the existing sketch-based methods [24, 25] derive the garment shape from the drawn sketch using a fairly simplistic shape interpretation which often leads to creation of unnatural looking garments (Figure 2 (left)) while others [3, 21] make restrictive assumptions about the user input. Most notably, they require users to provide the 3D locations for all the garment seams, a task that can be problematic for those lacking tailoring expertise.

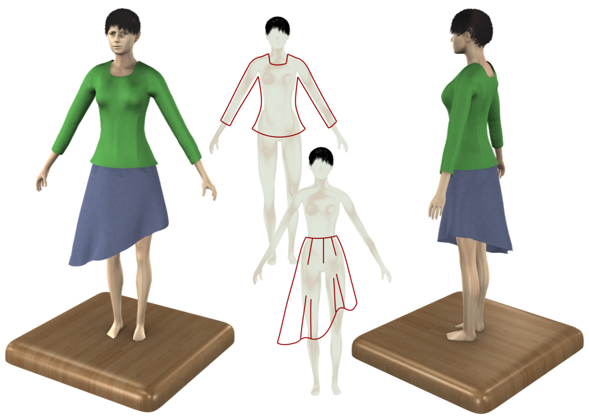


Figure 1: A believable outfit created from input sketches using our context-aware modeling system.

Our work aims to overcome these drawbacks while

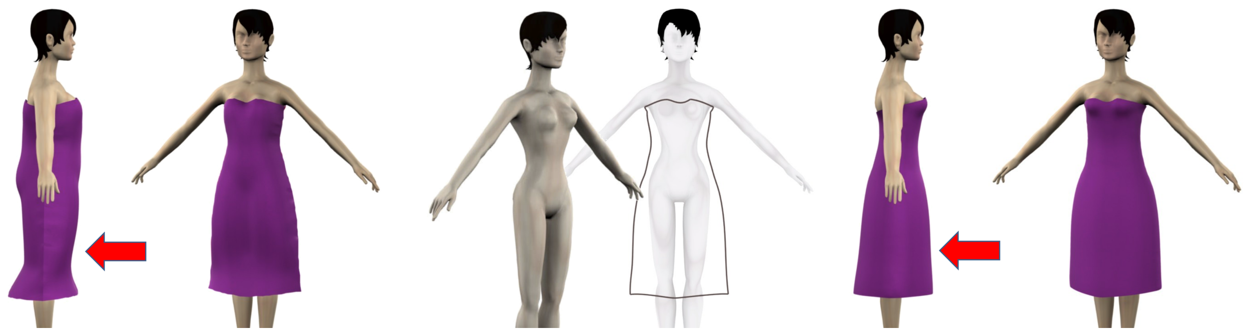


Figure 2: The dress created using an offset-based interpretation [24] (left) has an unnatural side profile shape, inconsistent with the input front-view sketch (middle), while our result (right) plausibly interprets user intent.

maintaining the ease of modeling provided by a sketching interface. We propose an alternative context-aware interpretation of garment sketches based on several observations about the factors that influence the shape of garments, such as garment cut, gravity, and contact with the body. Specifically, we speculate that on typical garments the same *context*, or combination of factors that determined the silhouette shape in the sketched view defines the silhouette shape in other views as well. By analyzing the factors that influenced the silhouettes depicted in the input sketches we predict the garment shape in other views. We use this prediction to guide a geometric modeling algorithm especially tailored for garment modeling, introducing a new method for generating mesh models that approximate generalized surfaces of revolution while satisfying interpolation and smoothness boundary constraints.

To validate our approach we generated a variety of garment models using the context-aware technique and performed a user study where viewers were asked to compare input sketch interpretations provided by our method to those created using earlier techniques. In the study, the participants consistently ranked our results as better reflecting the user intent (Section 5). As demonstrated by the study and the additional results included in the paper our method successfully creates garments that provide a believable interpretation of the user input.

We note that our goal is to create believable virtual garments, and not to reverse-engineer real garments that can be manufactured from planar patterns. Reverse-engineering would require the user to provide exact seam location, a challenging task for non-experts. More importantly, users may intentionally draw infeasible yet plausible garments (e.g. green dress in Figure 8, row three), which do not satisfy actual physics constraints, such as gravity, and which thus cannot be truly manufactured. Instead our work focuses on creating believable garment shapes from simple user sketches. Given a 3D garment model such as those created by our system methods such as [29, 27, 8] could potentially be used to obtain suitable patterns.

2. Related Work

Our work is closely related to previous approaches for garment modeling and draws on recent advances in sketch-based modeling, discussed below.

Garment Modeling: The traditional garment modeling pipeline used in commercial softwares, such as [10, 4, 15], follows a completely manual approach similar to real life garment design [5]. Users typically start by designing planar patterns and lifting those into 3D, a step that requires significant tailoring expertise. They then use physical simulation to obtain the desired realistic-looking results. The simulation often requires a significant amount of trial-and-error-based parameter tuning to obtain a desired look and, despite recent advances [6], simulation interfaces remain challenging to use for non-experts. The entire process is extremely labor intensive and difficult to master.

Consequently, numerous attempts were made to simplify the modeling process. For instance, [28, 2] use templates of clothing components and define rules on how those fit together to create complete garments, while [7] propose a user friendly interface for lifting patterns into 3D. However, such methods still rely on the user to do much of the traditional tailoring such as developing the component templates [28, 2] or providing the actual patterns [7].

Scanning of dressed humans [1, 26] provides virtual models of real garments but is not suitable for design of new, non-existent ones. Moreover, scanning requires specialized equipment, time, and know-how.

Turquin et al. [24] introduced a sketch based system for garment modeling drastically simplifying the modeling process. In this system users directly sketch lines on top of a 3D virtual mannequin, which are then used to create 3D garments that conform to the mannequin's shape. The lines drawn are labeled as silhouettes or borders. The garment is then modeled as an offset surface surrounding the mannequin with the distance to the body along the silhouettes used to determine the surface offset across the front and back. This approach creates fairly plausible results on tight garment sketches. However, on sketches of looser, more complex garments, the results look less believable and appear inconsistent with the input sketch (Figure 2, left). Turquin et al. [25] use the same offset-based sketch interpretation, but augment the set of lines that can be drawn, introducing special notations for complex hems and folds.

Recent methods [3, 21] use the observation that garments are piece-wise developable or nearly-developable to generate more believable results. Decaudin et al.[3] also add procedural folds to garments to increase realism. Both methods rely on the user to provide the garment seam lines which separate the different fabric panels and fail to provide meaningful results whenever the provided seams do not admit a developable [21] or near-developable [3] interpolant. The seam placement for complex garments is often unintuitive, e.g. see the back of the red dress in Figure 9, last row. In addition, both developability improvement algorithms ignore possible collisions with the mannequin, allowing for creation of garments that intersect it.

As demonstrated by the results (Section 5), our context-aware method provides a more realistic-looking interpretation of the input than [24, 25]. In contrast to [3, 21] it does not require the user to specify seam locations to generate believable results and provides a plausible resolution of body contacts.

Sketch-Based Modeling: Sketch-based techniques have become a popular tool for interactively generating 3D geometry. Some of the methods focus on generating smooth closed surfaces or combinations of those using fairly general assumptions, e.g. [13, 19]. Others target modeling of specific classes of objects, e.g. plants [17], stuffed animals [11], or buildings [12] using domain specific knowledge. A similar trend of using particular modeling contexts is noticeable in image-based modeling with examples including plants [18], hair [16], and architectural structures [22]. Our work follows a similar approach building a context-aware model for interpreting garment sketches.

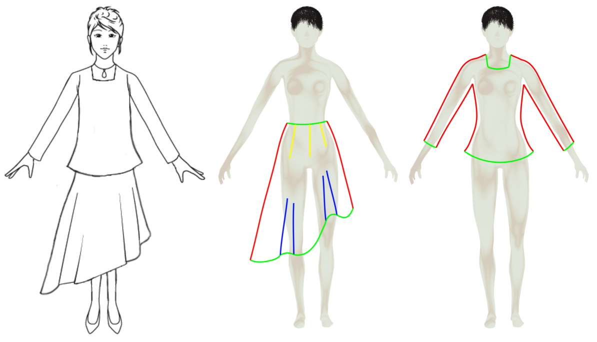


Figure 3: A typical traditional fashion drawing and annotated sketches of similar shirt and skirt drawn on top of a 3D mannequin (silhouettes colored in red, borders in green, folds in blue, wrinkles in yellow).

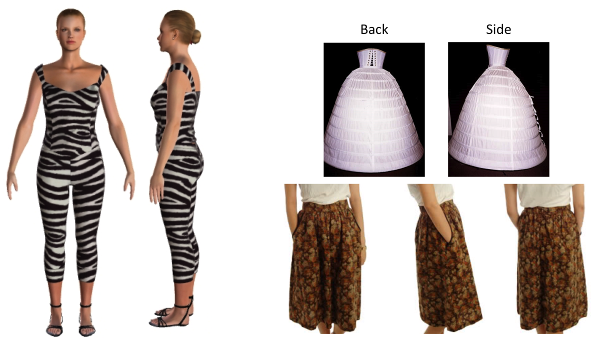


Figure 4: Typical correlation of garment silhouettes in front and side views: tight silhouettes in one view are indicative of tight silhouettes in others; Silhouette shape in loose area is largely similar in all views.

3. Context-Aware Sketch Interpretation

Given a front view garment sketch, e.g. Figure 3, left, humans appear to easily visualize the overall 3D shape of the drawn garment. Moreover, the interpretations seem largely consistent between different viewers. Thus, sketching provides an intuitive medium for conveying garment shape. To use a sketching interface for modeling the first challenge is to develop a suitable automatic interpretation of sketches that mimics the human one. In the past, supervised learning methods had been successfully applied for similar tasks, however using one would require a large database of sketches and corresponding 3D garments, which is currently unavailable. Instead, we base our interpretation on an analysis of the major factors affecting typical garment shapes and their impact on the garment features captured by the

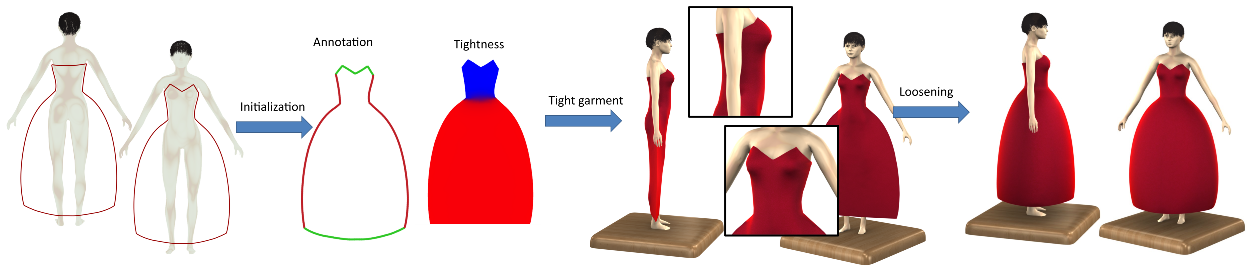


Figure 5: Algorithm overview (left to right): user input (front and back); annotated front view (silhouettes in red, borders in green); tightness function defined over the 2D surface (tight regions in blue and loose in red); tight "wrapper" garment; final output correctly modeling loose regions.

sketch.

The outline of the sketched garment provides the main cues regarding its shape [24]. It consists of two distinct types of curves: *silhouettes*, which represent the locations where the back and front garment surfaces coincide, and *borders* or garment edges (Figure 3, right red and green respectively). The silhouettes, especially when combined with a mannequin on which to fit the garment, appear to provide the strongest cues to the garment's 3D shape. The open question is how do humans interpret them, as the offset-based interpretation [24] seems inadequate (Figure 2). We speculate that the human perception of silhouettes is largely influenced by the *context* or factors that may have impacted the silhouette shape, and specifically the fact that *on typical garments the same factors that affect the shape of the side silhouettes seem to largely determine the garment silhouette shape in other views.*

We first observe that garments, or garment parts that are tightly fitting along the sides, are typically tightly fitting on the front and back as well. Thus, given tightly fitting side silhouettes one can expects the garment to be tight all around, with the shape of the garment across the back and front determined by the contact with the body (Figure 4, left).

We then note that without any additional constraints fabric hanging under gravity has vertical silhouettes. Thus, intuitively, one would expect the garment silhouettes in front or other views, once away from the wearer's body to be completely vertical. Non-vertical silhouettes in loose areas reflect a special tailoring effect achieved either through strategic seam placement or other means (Figure 4, top-right). In real garments, in our experience, these tailoring operations are often repeated across the front and back to obtain a similar silhouette shape in front and side views (Figure 4, right). Thus, one can view the garment surface in loose areas as a generalized surface of revolution, where the silhouette is the *profile* swept around the body. The sweep *trajectory* is determined by the tight areas where the garment comes in contact with the body. The transition between the tight and loose areas is typically smooth. Lastly, we observe that the side-view silhouettes are often slightly straighter than the front-view ones, as seams are more likely to be located off-center, and in seamless areas gravity causes the fabric to straighten, leading this generalized surface of revolution to be slightly "squashed".

To use these observations for modeling requires an algorithm for generating surfaces which tightly fit the mannequin in some areas and smoothly transition to a generalized surface of revolution in others (Section 4).

In addition to using silhouettes, viewers appear to derive some depth information from the shape of the sketched *hemlines*, or bottom edges of the garment. This interpretation is driven by a combination of two effects. First, garments are assumed to be drawn in a perspective view where the viewer is standing at a finite distance from the drawn figure and at the same eye level. Second, typical 3D garment hemlines are expected to be straight, i.e. planar and perpendicular to the (x, y) view plane. Thus viewers tend to perceive small variations in hemline height (Figures 3 and 5, left) as originating from the perspective, and thus indicating changes in depth. This depth inference is very approximate, as no accurate eye or camera positioning is used. Section 4.4 discusses how to use this subtle depth information to further augment the realism of the modeled garments. The hemline of the resulting garments, e.g. Figure 5, right, has a similar shape to the sketched one when rendered from a typical view point.

In addition to outlines, user-drawn sketches or fashion drawings often contain other lines that provide information about the garment shape, with folds and wrinkles being the two most common ones. Turquin et al.

[25] provide an elaborate interface allowing users to add a rich set of folds and wrinkles to the created garments. The setup can be used as-is to enrich garments created by our system in a post-processing step. In addition we provide a basic fold and wrinkle modeling tool directly within our system (Section 4.4). At the same time, since garments usually exhibit numerous wrinkles, expecting the user to sketch all of those is impractical, as it would require both time and artistic skill. Instead, procedural wrinkling [3, 20] can significantly increase garment believability with little to no user effort. We used this approach to wrinkle the shirts and pants in Figure 1.

4. Garment Surface Modeling

Our modeling algorithm generates three-dimensional garment surface meshes using as input the 3D mannequin on which we fit the garment and a set of sketched curves describing it. Following an initialization stage (Section 4.1) it computes the desired garment shape in two steps (Figure 5). It first generates a tight-fitting *wrapper* garment that provides a feasible geometry for the regions where the silhouettes are close to the mannequin (Section 4.2), and then updates the garment shape in loose regions while ensuring a smooth transition between the two (Section 4.3). It incorporates details such as hemlines, folds and wrinkles into the final garment shape as described in Section 4.4.

4.1. User Interface and Initialization

User Interface: We use a similar interface to [24] with the user drawing the garment outlines and other characteristic curves on top of a mannequin in front or back views (Figure 5, left). To support different front and back boundary shapes users can redraw boundaries in either view, while retaining the previously drawn silhouettes. The system then annotates the curves as silhouettes, boundaries, folds, or wrinkles (Figure 5, middle-left). It classifies outline curves as silhouettes if they do not cross the mannequin and as boundaries if they do. Interior curves are classified as folds if they touch the boundaries and as wrinkles otherwise.

Initialization: Garment hemlines are usually expected to be planar or nearly so, with the sketched hem shape reflecting the depth of the hemline vertices rather than their height in the view plane. Thus, our system identifies hemlines (as lower garment boundaries appropriately oriented), and straightens them out in the view plane to obtain their (x, y) coordinates (Figure 5, middle-left). The degree of straightening can be controlled by the user, the default being a perfectly straight hemline. The original hemlines are used to improve the final surface shape (Section 4.4).

Once all the lines are drawn and classified, the modeling process triangulates the 2D front and back outlines while conforming to the interior curves.

Assigning Local Coordinate Frames: The garment surface in loose regions roughly follows a generalized surface of revolution, with the axis of revolution corresponding to the associated body part. For a standard standing mannequin, the axis direction associated with much of the garment can be set to vertical, with the exception of sleeves where the axis corresponds to the direction of the appropriate upper/lower arm bones. Garment vertices are associated with the appropriate axis based on the closest point on the mannequin in 2D. Each vertex is then assigned a local coordinate frame that consists of a vector Y' aligned with the axis of revolution and two vectors X' and Z' orthogonal to it, with X' placed in the image plane. Note that with the exception of sleeves the local frame coincides with the global one.

Computing Tightness: Since the modeling differs significantly in tight and loose regions of the garment, our initialization computes a tightness function across the mesh surface which indicates how tight we expect the garment to be at any given point. The function is set to zero in loose regions and is close to one in tight ones (Figure 5), and will come into play when modeling loose regions (section 4.3). We first compute the tightness values along the silhouettes and then use these values to compute tightness across the front and back. The tightness at the silhouette vertices is initialized to a function of the planar distance from the vertex to the body (normalized with respect to the mannequin bounding box). We use a steep exponential falloff, as we want tightness to become zero once we move away from the mannequin even slightly. The user can modify the silhouette tightness values to account for sketching inaccuracies such as tight silhouettes drawn too far from the body.

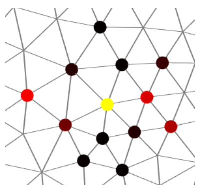
To distribute the values across the mesh we solve a least-squares minimization, which assigns similar tightness values τ to adjacent vertices, while holding the values along the silhouettes fixed:

$$\min \sum_{ij} w_{ij} (\tau_i - \tau_j)^2.$$

We aim to have smoothly changing tightness values across the front and back while avoiding oversmoothing along the silhouette direction. Thus the weights w_{ij} assigned to each pair of vertices depend on the orientation $N_{ij} = (v_i - v_j)/||v_i - v_j||$ of the vector $v_i v_j$ with respect to the local coordinate frame,

$$w_{ij} = e^{-\frac{|N_{ij} \cdot Y'_i|}{\sigma_1}^2} \quad (1)$$

Based on experiments we set $\sigma_1 = 0.5$. To reduce the solution's dependency on the triangulation, the sum $\sum_{ij}$ includes a two-ring neighborhood of each vertex.



In the example on the right, the neighbors of the yellow vertex i are colored black to red based on the weight w_{ij} (low to high). The axis direction in this example is vertical. We solve the resulting linear system using Cholmod [23].

4.2. Tight Garment Computation

We now proceed to compute the actual surface geometry, assigning depth, or z, values for the mesh vertices. We start by computing a garment which is tight everywhere, creating a plausible solution for the regions classified as tight. While there are numerous spring-system approaches for generating such tight garments [5], we found that the following modified Laplacian formulation provides adequate results in our setup at a reasonable computational cost.

A basic zero length spring, or Laplacian, system with standard boundary conditions implicitly minimizes the mean curvature of the surface. Since garments are nearly developable piece-wise we would like the surfaces we obtain to have small Gaussian curvature, possibly at the expense of higher mean one. We observe that one way to reduce the Gaussian curvature is to minimize the squared normal curvature on the surface in a consistent direction. For garments, the local axis of revolution provides a natural choice. Thus to obtain a plausible wrapper surface we use a weighted Laplacian formulation which aims to minimize the curvature of the surface, intuitively searching for a minimal surface that interpolates the sketch outline and doesn't intersect the mannequin. The choice of weights prioritizes normal curvature minimization (vertex position similarity) along the local axis Y':

$$\min \sum_i ||v_i - \frac{1}{\sum_{ij} \phi_{ij}} \sum_{ij} \phi_{ij} v_j||^2, \quad (2)$$
$$\phi_{ij} = (1-\gamma) + \gamma e^{-(|N_{ij} \cdot X'_i|/\sigma_2)^2}$$

where v_i are the mesh vertices and $\sigma_2 = 0.1$. The impact of the developability factor γ is shown in Figure 6. The main difference is in the amount of garment "stickiness" in concave regions, such as the small of the back, where using the default value of $\gamma = 0.5$ we achieve results more consistent with less stretchy fabrics.

The resulting energy term is combined with the requirements to preserve the known 2D vertex positions along outlines and to prevent garment intersection with the body. Intersection avoidance is incorporated explicitly through the use of a Gauss-Seidel type solver. The process is initiated by moving each interior vertex of the mesh along the depth axis to avoid intersection with the mannequin. At each iteration, if the new position computed based on Equation 2 is inside the mannequin, the vertex is only moved along the vector between the old and new positions until it lies on the mannequin surface. After the process converges, to improve the quality of the resulting mesh we apply a few iterations of standard tangential smoothing.

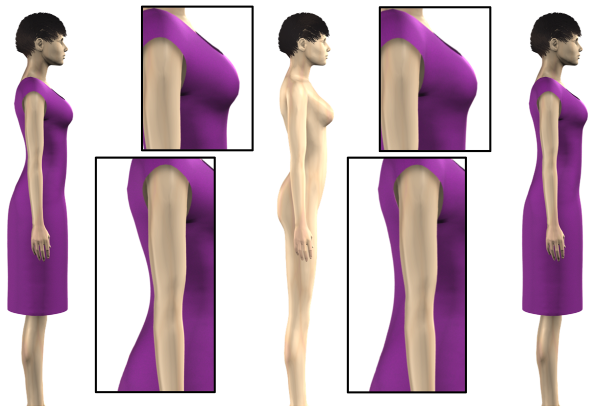


Figure 6: Tight purple dress (Figure 10) generated with developability term $\gamma = 0$ (left) and $\gamma = 0.5$ (right) (side-view of the mannequin used shown in the center).The difference is most noticeable in the concave areas.

4.3. Letting the Garments Loose

For a tight-fitting outfit the computed surface provides a feasible interpretation, however outfits that have loose regions require further processing. Specifically, we want the garment in these areas to form a generalized surface of revolution. Since we also require a smooth transition between this surface and the one generated for tight regions standard methods for generating swept surfaces are not suitable as-is for our task.

Instead, the following two key observations lead us to adopt a normal-based modeling strategy described below. We first note that along any given sweep trajectory on a generalized surface of revolution the component of the surface normal aligned with the axis of revolution remains constant while the two other, orthogonal, components defined in our local frame change smoothly. In contrast, along a profile, the two orthogonal components of the normal remain constant (up to a normalization factor), while the *axis-aligned* one changes.

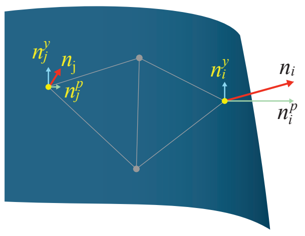
A typical sketch contains two opposite silhouettes for each loose region, with the vertex normals well defined along each one (the depth component of each normal is zero). Therefore, to extract a surface of revolution from such silhouettes one can interpolate these normals over the front and back, separating the axis-aligned normal component from the two orthogonal ones. Specifically, the axis-aligned components should remain very similar along each trajectory while the orthogonal components should be very similar along a profile. Such interpolation of two opposite silhouettes effectively defines two half-circle surfaces of revolution (each determined by a choice of depth direction when interpolating the two axis-orthogonal components of the normal). Enforcing normal preservation in tight regions, while preserving normal continuity as described below changes the surface trajectory to one that better follows the mannequin body shape. The final surface geometry is then generated by computing vertex positions that satisfy the normals.

4.3.1. *Solving for Normals*

Based on the requirements above we compute the normals across the loose areas of the garment surface using a weighted Laplacian-type functional which consists of three terms: preservation of wrapper normals, smoothness or similarity of adjacent axis-aligned normal components predominantly enforced along the trajectories, and similarity of orthogonal components predominantly enforced along the profiles. The 2D normals of the sketched silhouettes serve as boundary conditions.

$$\min \sum_i \frac{\tau_i}{\psi} \|n_i - n'_i\|^2 \qquad (3)$$
$$+(1-\frac{\tau_i}{\psi}) \sum_{ij} [w_{ij} \|n^y_i - n^y_j\|^2 + w'_{ij} \|n^p_i - n^p_j\|^2]$$

n^y_i is the Y' component of the normal in the local coordinate frame, n^p_i is the orthogonal component, and n'_i is the initial normal computed on the wrapper surface. The weights w_{ij} are the same as for the tightness computation (Equation 1), giving higher weights to axis-aligned normal components lying on the same trajectory . The weights w'_{ij} are similarly defined with respect to the X' axis $w'_{ij} = e^{-(\frac{|N_{ij} \cdot X'{}_i|}{\sigma_1})^2}$ to obtain more similar orthogonal components along each profile.



In the figure on the right, Y' is the vertical direction and hence the weight w_{ij} will be high, forcing n^y_i and n^y_j to be very similar. In contrast, the weight w'_{ij} will be small, allowing the orthogonal component of the two normals to diverge. As in the tightness computation, the sum $\sum_{ij}$ includes a two ring neighborhood of each vertex. The parameter ψ controls the impact of the wrapper normals on the solution. In loose regions these normals are nearly horizontal, thus adding them into the framework creates the side-view straightening effect discussed in Section 3. However, we want this impact to be very small, thus by default we use a very high value of $\psi = 5000$.

The function above has to be combined with a normalization constraint forcing normals to be of unit length. Since in our setup we expect the axis-aligned component of the normal n^y to be interpolated separately, the renormalization is applied only to the orthogonal components n^p. The combined formulation is challenging to solve using direct solvers. At the same time since the rotations introduced are fairly large, using an unconstrained linear solver and then renormalizing the results as done in many linear deformation setups, can in our experience introduce visible artifacts. However, we found that the basic Gauss-Seidel scheme converges sufficiently fast for our needs using the wrapper surface normals as an initial guess.

4.3.2. *Solving for Positions*

Given the normals we search for the vertex positions that satisfy them. In general this is an ill-posed problem, as numerous solutions exist. Thus we need to introduce per-vertex constraints to stabilize it. Since the final solution can significantly deviate from the tight wrapper surface, pulling the solution toward the initial-positions [14] provides unnatural results. Enforcing mesh smoothness by introducing a Laplacian term provides better results, positioning the surfaces more or less where expected, but tends to over-smooth them.

To obtain the desirable result, we first solve for positions that satisfy the normals while preserving surface smoothness and then recompute the positions replacing the smoothness term with one preserving the positions computed in the first step. Specifically we first minimize $P(v) + R_1(v)$, where

$$P(v) = \sum_i \sum_{(i,j,k)\in T} (n_i \cdot (v_j - v_k))^2$$

$$R_1(V) = \mu_1 \sum_i (v_i - \frac{1}{|(i,j)|} \sum_{(i,j)} v_j)^2$$

and T is the set of mesh triangles. $P(v)$ is a quadratic functional expressing normal preservation proposed in [14]. R_1 enforces both smoothness and triangle shape by aiming to place each vertex at the average of its neighbors, without necessarily collapsing the mesh. To avoid distorting the solution in tight regions, the sum in R_1 iterates only on vertices with tightness values smaller than 1. We found that a high value of $\mu_1 = 20$ is necessary to stabilize the solution. In the second step we replace R_1 with $R_2 = \mu_2 \sum_i (v_i - v_i'')^2$, where v_i'' are the positions from the first solution step and $\mu_2 = 1$.

The boundary conditions for both steps are: preservation of positions in tight regions with $2\tau_i$ as the per vertex weight, and preservation of silhouette and boundary positions with weights of 7 and 1 respectively (slight boundary displacement allowed by the smaller weight tends to increase realism). Both steps use Cholmod [23] to solve the corresponding linear systems. In some rare cases the obtained surfaces can intersect the mannequin, as collisions are not prohibited explicitly. In such cases we first resolve the collisions, pushing the colliding vertices outside and then perform a few Gauss-Seidel iterations to correct the shape in the surrounding area. At the end of this step we have a believable overall garment shape consistent with the drawn outline.

4.4. Adding details

Additional shape information provided by hemlines, folds and wrinkles is incorporated into the system using the following modifications (Figure 7).

Hemlines: We use the depth information provided by the hemline shape to improve the shape of the garment. Given the 2D sketch and the location of the viewer, and assuming that the hemline is straight, i.e. embedded in a plane orthogonal to the view plane, it is possible to compute the three-dimensional positions of hemline vertices by inverting an estimated perspective matrix. This process can also determine the (x, z) component of the surface normals along the hemline. Conversely, given the 3D position of a vertex, we can estimate the viewer location. Our method first computes a least-squares estimate of viewer location,

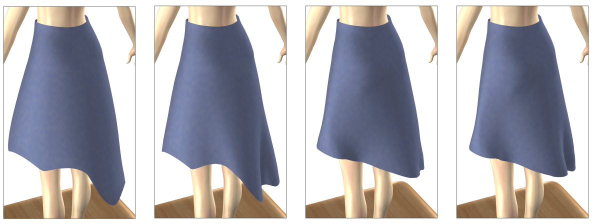


Figure 7: Adding details to the skirt in Figure 1, back view (left to right): modeled using only silhouette information; with folds but without using hemline information; with hemline shape taken into account, no folds; and with both.

based on vertex positions as determined in the previous section. We then use this estimate to calculate new target normals along the hemline. Finally, solving for optimal normals and positions throughout the garment, we obtain a new model with similar overall shape that better conforms to the sketched hemline. Since not all sketches use perspective (e.g. Figure 10, top-right), we abort the hemline computation if the new positions intersect the mannequin or drastically differ from the original. The hemline incorporation not only helps to resolve unnatural uneven hems created when perspective drawing is not corrected, most notable in Figure 9 rows two and three left, but helps accentuate folds when combined with the next step.

Folds: After computing the basic normals across the surface, we incorporate folds into the setting as lines along which the normal is rotated away from the view direction, perpendicular to the fold axis. The direction of the fold is set consistent with the previously solved normal direction at its bottom point and the default rotation angle places the normal at this point into the view plane (consistent with the expectation of folds as interior silhouettes). Both the direction and the angle can be adjusted by the user if desired. Starting at the bottom point of each fold, we rotate the normal as specified and propagate the rotation upward in a smooth manner, where the normal at the top of each fold is left unchanged. We then re-solve for the surface normals, keeping the fold normals fixed.

Wrinkles: While wrinkles only slightly change the shape of a garment, due to their ubiquity, garments lacking them look unnatural. To add wrinkles we smoothly indent the surface along the wrinkle line. The user can specify the wrinkle direction, with the default being outward, like for most real-life wrinkles. Additional wrin-

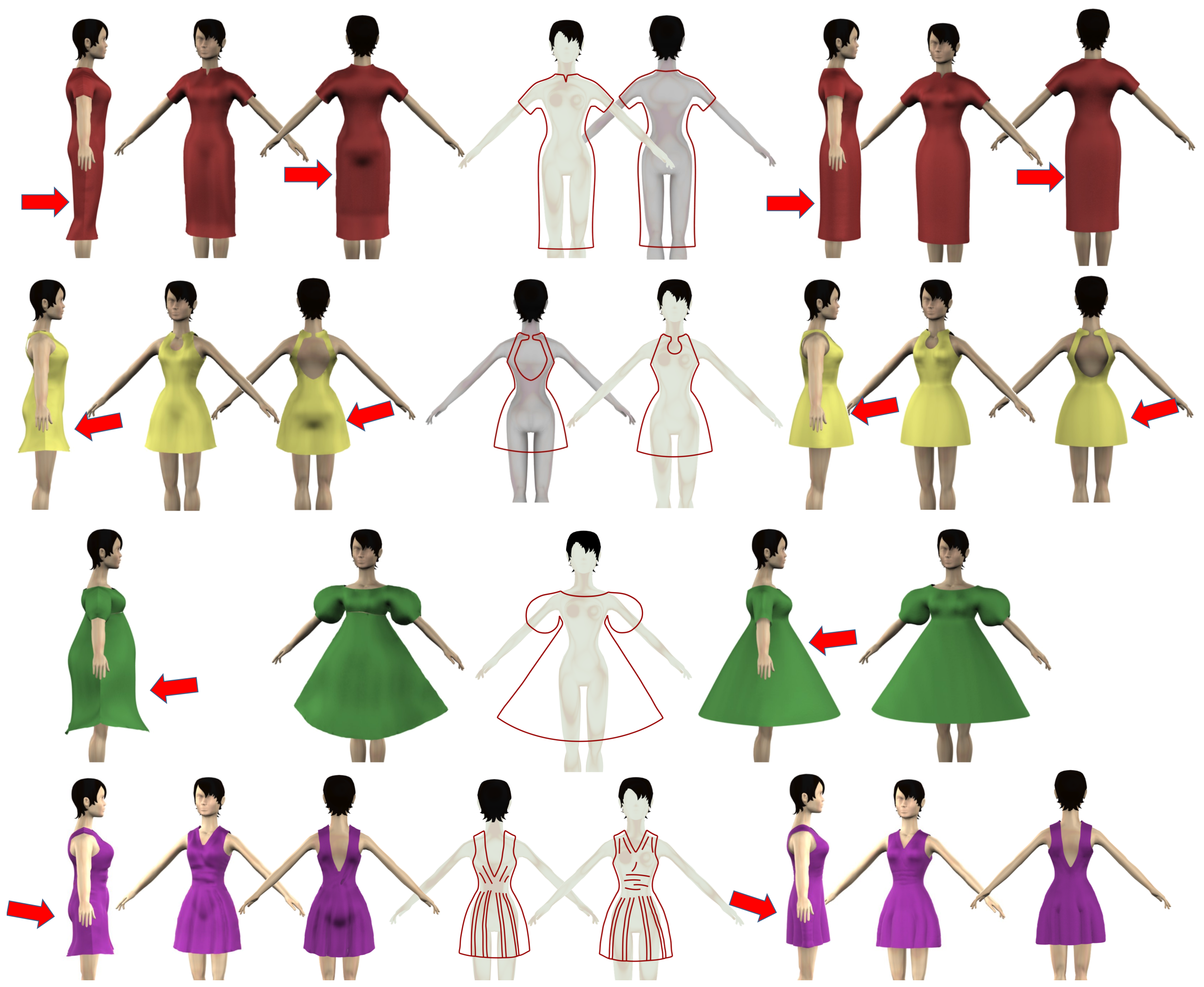


Figure 8: Comparisons with [24] (top three rows) and [25] (bottom). Previous result on the left and ours on the right. The mannequin used is the same as in Figure 2.

kles can be added using a procedural approach [3, 20].

With the extra details added-in we obtain a believable garment consistent with the user sketch (Figure 1).

5. Results and Discussion

We have tested our method on over twenty inputs shown throughout the paper and in the accompanying video. Those include tight-fitting (purple dress in Figure 10, red dress in Figure 8) as well as puffy or loose garments (skirt in Figure 1, ball-gown in Figure 5); dresses, skirts, shirts, and pants; long and short sleeves; and different female and male mannequins. We included both models created from inputs used by previous methods (Figures 2,8,9) as well as totally new ones (Figures 1,5,10). Two of the new inputs (Figures 1,5) were drawn by an artist and the others were drawn by students with no artistic training. In all these examples our modeling system was able to create realistic-looking garments which provide a plausible interpretation of the input sketches.

Comparison and User Study: To get an unbiased evaluation of the results we conducted a web user study, where a random selection of 65 participants were asked to compare our results to those of previous methods in terms of believability and fidelity with respect to user input. Participants were presented with an input sketch, a mannequin, and two modeled outfits (in front and side views) side by side, and were asked to answer which

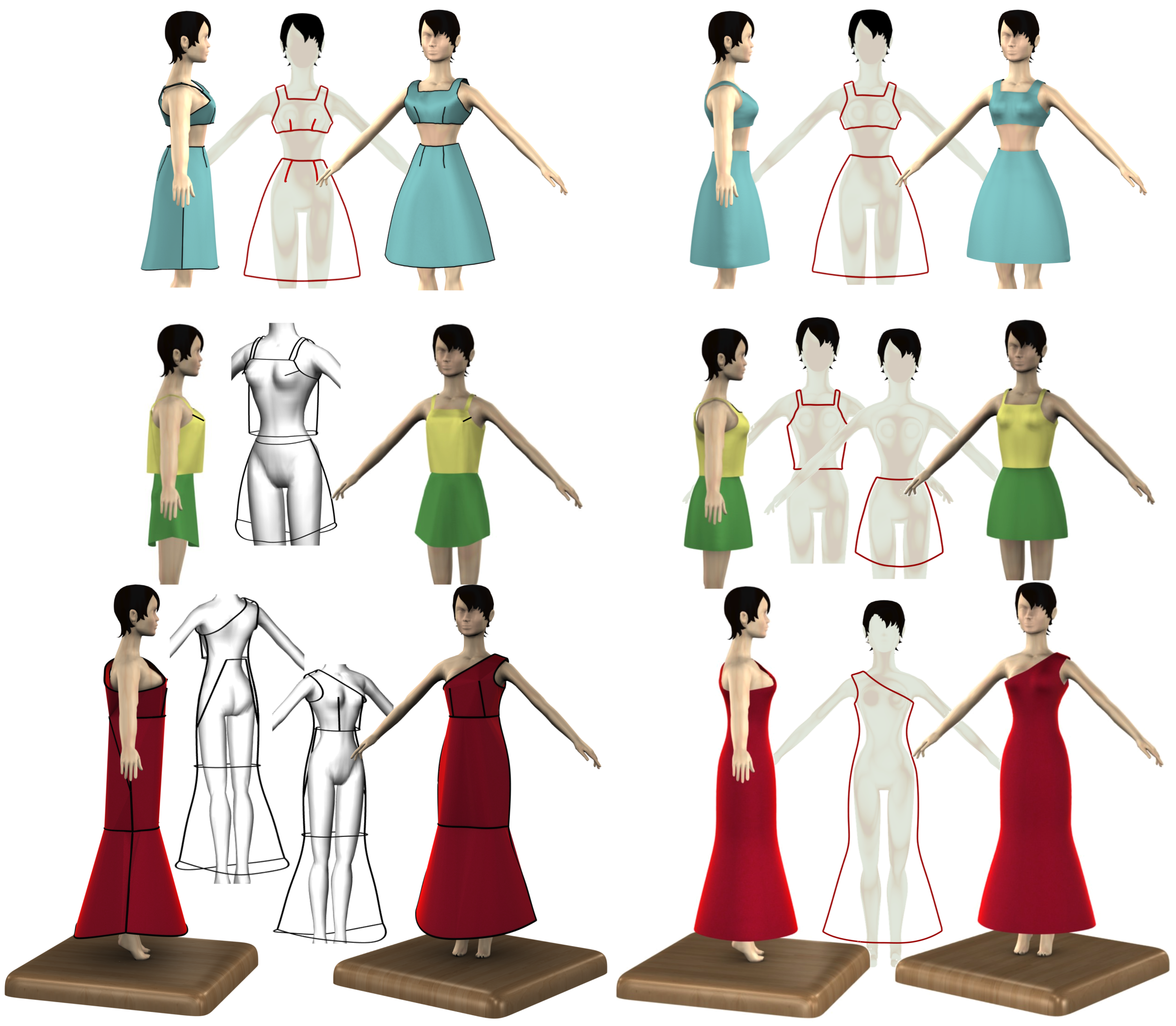


Figure 9: Comparisons with [3] (top) and [21] (bottom two rows). The inputs to both methods include extra 3D seams. Even for basic models such as skirts and tops [3, 21] require the user to specify darts to produce meaningful results.

garment looks like a more believable realization of the sketch. The users were not provided with any special technical instructions. The questions were presented in random order to avoid bias. The survey included twelve outfits, seven from [24], one from [25], two from [3], and two from [21] (all provided to us by the original authors). Overall our results were preferred in 80% of the comparisons (616/770). When considering only the methods of [24, 25], where the input is similar to ours, the number increased to 84%. The only input for which less than two thirds of the participants preferred our results was the top and skirt outfit (Figure 9, middle) where the vote was equally split between our result and that of [21]. Note however that both [21] and [3] require users to specify additional seams even for such simple inputs. Specifically, both require user specified darts on tops and skirts. Since both methods ignore collisions the resulting garments can intersect the mannequin, thus for instance we had to manually scale the bra in Figure 9, top-left to resolve collisions. Overall, the study validates our claim that the context-aware method we propose provides more realistic interpretation of user sketches, without requiring user-specified seams.

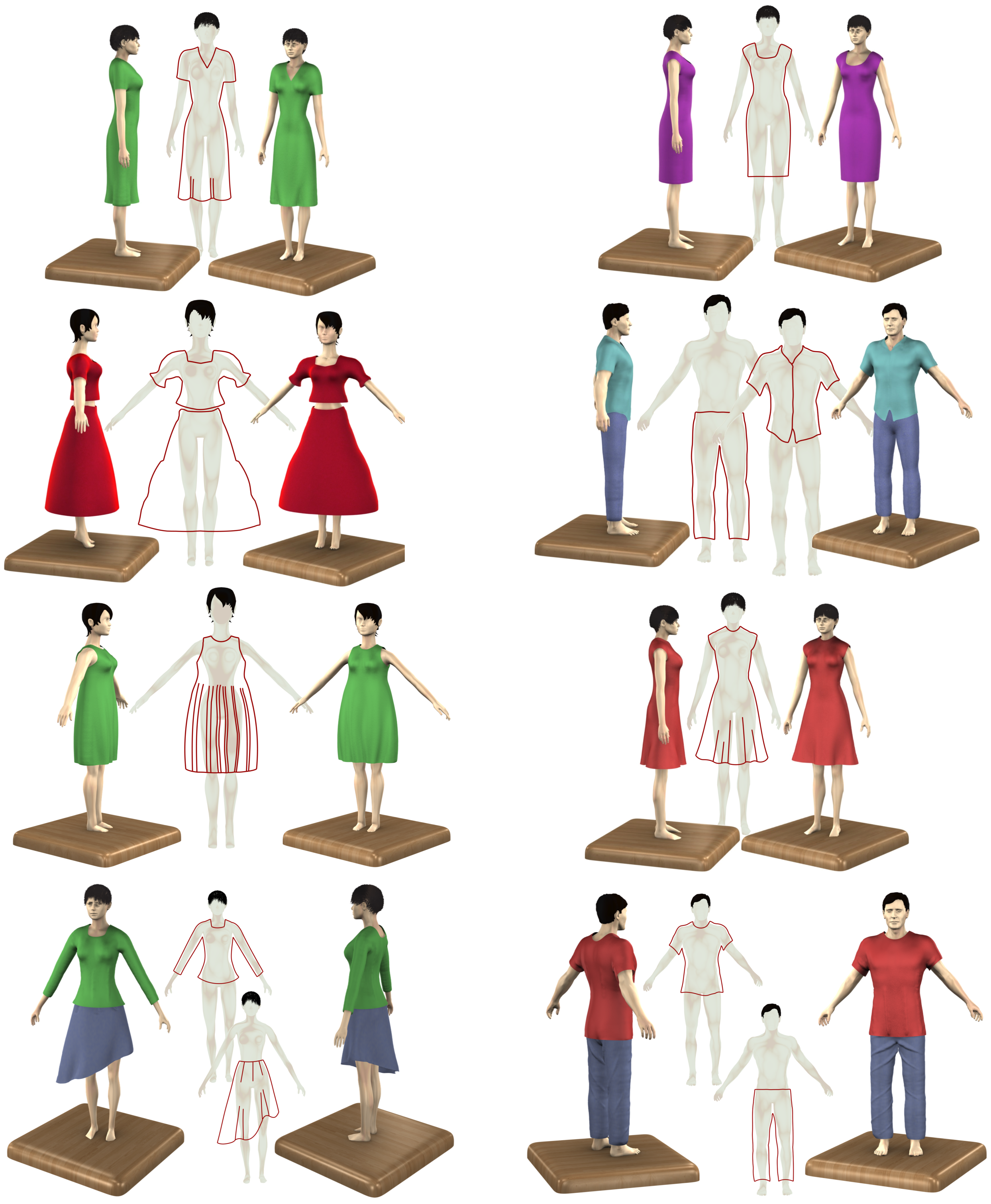


Figure 10: Diverse garment models generated by our system using different mannequins.

Parameter Setting: The majority of the coefficients used in our modeling system are fixed for all examples and we expect the specified default values to be applicable to all models.

However, there are three parameters which we found users may want to control: developability γ (Equation 2), straightening ψ (Equation 3), and silhouette tightness (Section 4.1). The first is strongly linked to the fabric and cut of the modeled garment, which can vary from model to model. For instance, for the red "chinoise" dress (Figure 8 top) we turned it off, as we wanted the dress to be very fitted. Figure 6 shows the impact of the developability factor on the purple dress (Figure 10). ψ controls how straight the garment is in a side view compared to the front one. The sketch provides little to no cues in this regard, hence the "correct" output is very subjective. Lastly, the silhouette tightness falloff can vary based on a user's drawing style (e.g. in Figure 2 even the supposedly tight silhouettes are drawn quite far).

Performance: Our unoptimized garment modeling code takes 7 to 12 seconds to generate a garment with ten to fifteen thousand triangles in our testing environment (utilizing one processor on a 2.13 GHz Intel Core 2 workstation). Given that we are concerned with modeling the overall garment geometry this resolution was sufficient for all the models shown in the paper. The time is split between the global linear solves (tightness computation and global position solve, Section 4.3.2), which take 5-7 seconds; iterative solves (Section 4.2 and Section 4.3) 1-3 seconds, and collision queries 1-3 seconds. We believe that the linear solves which dominate the cost can likely be sped up using a tailored preconditioned iterative solver and the collision queries can be sped up by using fast GPU collision testing. The runtime depends on both the mesh resolution and the width of the loose garment areas, with the iterative solution steps converging more slowly for looser garments like the dress in Figure 5. The method does not depend on the mannequin complexity as the mannequin is only referenced via collision queries. Our implementation uses the Closest Point Transform [9] to reduce the cost of these queries and eliminate the dependency on the resolution of the mannequin.

The overall time for a modeling session can vary from ten minutes to half an hour depending on the garment complexity and the user's sketching abilities. These session times are significantly lower than those required by traditional garment modeling environments.

6. Conclusions

We presented a new system for sketch-based garment modeling than creates believable garments plausibly reflecting the user intent. The system is based on a context-aware interpretation of garment sketches combined with a suitable specialized geometric modeling mechanism, which allows for construction of generalized surfaces of revolution for the loose regions while maintaining smooth transition with the body-fitting wrappers surfaces in the tight areas.

Our method has some limitations, which can be addressed by future work. To create optimal results the current system relies on a couple of parameters which frequently require adjustment by the user, such as tightness and developability. One possible approach to minimize manual parameter adjustment is to introduce a machine learning setup which learns optimal parameters by analyzing manually-set ones for a variety of inputs. The quality of our results depends on the mesh triangulation used, with higher quality achieved on finer meshes, better aligned with the local coordinate frames. A built-in adaptive re-meshing scheme can therefore improve the quality of the obtained results.

Other possible areas of future research include direct design from existing fashion drawings skipping the user-interface step, and reverse engineering of garments where the goal is to find as-plausible-as-possible set of patterns for a given sketch or 3D garment.

Acknowledgements

We thank Pushkar Joshi and Vladislav Kraevoy for their help. This project was partially funded by NSERC, MITACS, and GRAND NCE.

References

[1] Bradley, D., Popa, T., Sheffer, A., Heidrich, W., Boubekeur, T., 2008. Markerless garment capture. ACM Trans. Graphics (Proc. SIGGRAPH) 27 (3), 99.

[2] Cochenile Design Studio, 2010. Http://www.cochenille.com.

[3] Decaudin, P., Julius, D., Wither, J., Boissieux, L., Sheffer, A., Cani, M.-P., 2006. Virtual garments: A fully geometric approach for clothing design. Comput. Graph. Forum (Proc. Eurographics'06) 25 (3), 625–634.

[4] Haute Couture 3D, 2010. http://www.gcldistribution.com/en/haute_couture_3d.html.

[5] House, D. H., Breen, D. E. (Eds.), 2000. Cloth modeling and animation. A. K. Peters, Ltd., Natick, MA, USA.

[6] Igarashi, T., Hughes, J. F., 2002. Clothing manipulation. In: Proc. UIST"02. pp. 91–100.

[7] Igarashi, T., Mitani, J., 2010. Apparent layer operations for the manipulation of deformable objects. ACM Trans. Graph. 29 (4).

[8] Igarashi, Y., Igarashi, T., 2009. Designing plush toys with a computer. Commun. ACM 52, 81–88.

[9] Mauch, S., 2009. Closest point transform. http://www.cacr.caltech.edu/ sean/projects/stlib/html.

[10] MayaCloth, 2010. http://caad.arch.ethz.ch/info/maya/manual/ MayaCloth.

[11] Mori, Y., Igarashi, T., 2007. Plushie: an interactive design system for plush toys. In: Proc. SIGGRAPH'07. New York, NY, USA, pp. 45–54.

[12] Müller, P., Wonka, P., Haegler, S., Ulmer, A., Van Gool, L., 2006. Procedural modeling of buildings. ACM Trans. Graph. 25 (3), 614–623.

[13] Nealen, A., Igarashi, T., Sorkine, O., Alexa, M., 2007. Fibermesh: designing freeform surfaces with 3d curves. ACM Trans. Graph. 26 (3), 41.

[14] Nehab, D., Rusinkiewicz, S., Davis, J., Ramamoorthi, R., 2005. Efficiently combining positions and normals for precise 3D geometry. ACM Transactions on Graphics (Proc. of ACM SIGGRAPH 2005) 24 (3).

[15] Optitex, 2010. http://www.optitex.com.

[16] Paris, S., Chang, W., Kozhushnyan, O. I., Jarosz, W., Matusik, W., Zwicker, M., Durand, F., 2008. Hair photobooth: geometric and photometric acquisition of real hairstyles. In: Proc. SIGGRAPH '08. ACM, pp. 1–9.

[17] Prusinkiewicz, P., Lindenmayer, A., 1990. The algorithmic beauty of plants. Springer-Verlag New York, Inc.

[18] Quan, L., Tan, P., Zeng, G., Yuan, L., Wang, J., Kang, S. B., 2006. Image-based plant modeling. ACM Trans. Graph. 25 (3), 599–604.

[19] Rivers, A., Durand, F., Igarashi, T., 2010. 3d modeling with silhouettes. ACM Trans. Graph. 29 (4).

[20] Rohmer, D., Popa, T., Cani, M.-P., Hahmann, S., Sheffer, A., 2010. Animation wrinkling: Augmenting coarse cloth simulations with realistic-looking wrinkles. ACM Trans. Graph. 29 (5).

[21] Rose, K., Sheffer, A., Wither, J., Cani, M.-P., Thibert, B., 2007. Developable surfaces from arbitrary sketched boundaries. In: Proc. Eurographics Symposium on Geometry Processing.

[22] Sinha, S. N., Steedly, D., Szeliski, R., Agrawala, M., Pollefeys, M., 2008. Interactive 3d architectural modeling from unordered photo collections. ACM Trans. Graph. 27 (5), 1–10.

[23] Toledo, S., Chen, D., Rotkin, V., 2003. Taucs: A library of sparse linear solvers. Http://www.tau.ac.il/ stoledo/taucs/.

[24] Turquin, E., Cani, M.-P., Hughes, J. F., 2004. Sketching garments for virtual characters . In: Proc. Eurographics Workshop on Sketch-Based Interfaces and Modeling. pp. 175–182.

[25] Turquin, E., Wither, J., Boissieux, L., Cani, M.-P., Hughes, J. F., 2007. A sketch-based interface for clothing virtual characters. IEEE Comput. Graph. Appl. 27 (1), 72–81.

[26] Vlasic, D., Peers, P., Baran, I., Debevec, P., Popović, J., Rusinkiewicz, S., Matusik, W., 2009. Dynamic shape capture using multi-view photometric stereo. ACM Trans. Graph. 28 (5).

[27] Wang, C., 2008. Computing length-preserved free boundary for quasi-developable mesh segmentation. IEEE Transactions on Visualization and Computer Graphics 14, 25–36.

[28] Wang, C., Wang, Y., Yuen, M., 2003. Feature based 3d garment design through 2d sketches. Computer-Aided Design 35, 659–672.

[29] Wang, C. C. L., 2008. Towards flattenable mesh surfaces. Comput. Aided Des. 40, 109–122.