EUROGRAPHICS Workshop on Sketch-Based Interfaces and Modeling (2004)
John F. Hughes and Joaquim A. Jorge (Guest Editors)

Sketching garments for virtual characters

Emmanuel Turquin[1], Marie-Paule Cani[1] and John F. Hughes[2]

[1] Laboratoire GRAVIR (CNRS, INP Grenoble, INRIA, UJF)
[2] Brown University

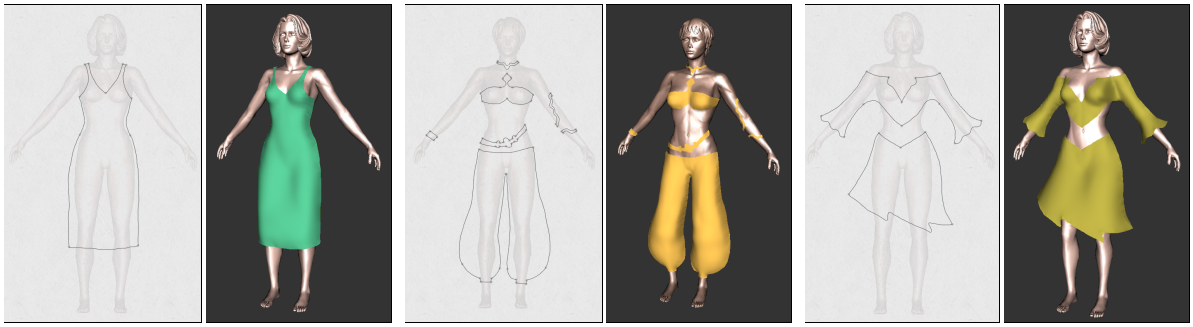

Figure 1: *Three examples of garments of various styles, respectively generated from the sketch on their left using our approach.*

Abstract

We present a method for simply and interactively creating basic garments for dressing virtual characters in applications like video games. The user draws an outline of the front or back of the garment, and the system makes reasonable geometric inferences about the overall shape of the garment (ignoring constraints arising from physics and from the material of the garment). Thus both the garment's shape and the way the character is wearing it are determined at once. We use the distance from the 2D garment silhouette to the character model to infer the variations of the distance between the remainder of the garment and the character in 3D. The garment surface is generated from the silhouette and border lines and this varying distance information, thanks to a data-structure that stores the distance field to the character's body. This method is integrated in an interactive system in which the user sketches the garment over the 3D model of the character. Our results show that the system can be used to create both standard clothes (skirts, shirts) and other garments that may be worn in a variety of ways (scarves, panchos).
Key words; Sketch-based interfaces, virtual garment, shape modeling.

Categories and Subject Descriptors (according to ACM CCS):
I.3.5 [Computational Geometry and Object Modeling]: Modeling packages
I.3.6 [Methodology and Techniques]: Interaction techniques

1. Introduction

Nowadays, 3D virtual characters are everywhere: in video games, feature films, TV shows... Obviously, these characters need clothing: when we encounter a person, we instantly notice his or her clothes, and feel that these tell us something about the person. Thus clothing on virtual characters provides a way to easily advance the story being told to the viewer.

The range of approaches used for clothing virtual characters is large: for incidental characters, the clothing may be no more than a texture map. For lead characters in feature films, full-fledged physical simulation of detailed cloth mod-

els may be used. And in the midrange, simple skinning techniques, combined with texture mapping, are common, providing some deformation of clothing as the character moves, but no physical realism. There are three problems one can associate with clothing virtual characters: the design of the clothes (*tailoring*), placing them on the character (*dressing*), and making them look physically correct (typically through *simulation*).

The process of tailoring, for human beings, involves choosing cloth and fitting it to the body, often making adjustments in the patterns of the cloth to adapt it to the particular person's form, and then sewing it. For virtual characters, clothing often has no "patterns" from which it is sewn, instead being represented by a simple polygon mesh that's constructed to fit the body. It's currently tedious to construct such meshes even without the issues of patterns and stitching. It's sometimes done by directly incorporating the cloth mesh into a character's geometric model, so that the character doesn't actually have *legs*, for instance, but just *pants*. In this case physical simulation is no longer a possibility, and when a character needs new clothes, it must be largely remodeled. An alternative approach involves drawing pattern pieces for a garment and positioning them over the naked form of the character, defining stitching constraints, etc. This can be tedious, especially when the character is not important enough to merit this level of effort; it also requires an understanding of how cloth fits over forms, although the actual pattern-and-stitching information may not be relevant after the tailoring is completed (except in the rare case where the physical properties of the cloth — was it cut on the bias? Does the cloth resist folding along one axis? — are later used in a full-fledged physical simulation).

Our approach combines tailoring and dressing into a single step to create a mesh that's suitable for later simulation or skinning approaches. The idea is to make it easy to generate simple garments that are adapted to an existing model. We believe that most people know better how to draw garments than the patterns which are needed to sew them. The aim of this work is thus to explore the use of a sketch-based interface for quickly constructing 3D virtual garments over a character model. This paper describes simple solutions to the problems of *shape generation* and *placement* of the clothing. The resulting system is so easy to use that it takes only minutes to create a simple garment.

1.1. Related work

Current interactive systems [HM90, WMTT93, BGR02] for designing garments and dressing virtual actors with them can be quite tedious: typically the user must draw each pattern piece on a planar virtual cloth, specify the edges to be stitched together, position the pieces around the virtual actor, and then finally run a simulation to obtain a convincing rest shape for the garment around the character model.

Sketch-based modeling systems such as [ZHH96, EyHBE97, IMT99, FBSS04] have became popular for interactively generating 3D geometry from sketches. Most of them only generate solid objects, as opposed to such surfaces as garments.

Two works have combined the idea of sketching with the goal of designing clothes: Bourguignon [BCD01] provided a 3D sketching method and used it to design garments over virtual actors. The sketch could be viewed from arbitrary viewing angles, but no 3D surface was reconstructed for the garment. Igarashi [IH02] described a sketch-based method for positioning garment patterns over a 3D body, but the user could not use the system for directly sketching the desired garment and still must know which pattern shapes will result in the garment he desires. That is to say, they addressed the *dressing* problem through sketching, but not the *tailoring* problem.

1.2. Overview

This paper presents a method for reconstructing the 3D geometry and placement of garments from a 2D sketch. As in [BCD01], the user sketches the garment directly on the 3D virtual actor body model. However, our method outputs a full 3D geometry for the garment, using the distance from the 2D garment silhouette to the character body model to infer the variations of the distance between the garment and the character in 3D.

More precisely, the reconstruction is performed in 3 steps: the lines of the 2D drawing are first automatically classified either as *silhouettes* (lines that do not cross the body) or as *border lines* (lines that do cross the body). A distance-to-the-body value is computed for each point of a silhouette segment and these distances are then used to determine desired point-to-body distances for the border lines. This distance information is then propagated in 2D to find desired point-to-body distances, which are then in turn used to determine the 3D position of the garment.

2. Sketch-based interface

2.1. Typical user interaction

To give a sense of the system's performance, we describe a typical interaction, in which a user sketches a skirt on a female model. The user first draws a line across the waist (see figure 2 (left)), indicating the top of the skirt, and then a line down the side, indicating the silhouette of the skirt, then a line across the bottom in a vee-shape indicating that he wants the front of the skirt to dip down, and finally the last side. A simple corner-detection process is applied to break the sketch into parts; one extra corner is detected by accident (at the bottom of the vee) and the user can delete it with a deletion gesture. He could also add new breakpoints as well, but none are necessary. Breakpoints play an important role in the 3D positioning process, since they determine

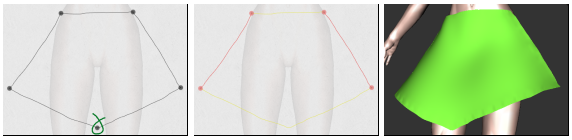

Figure 2: ***(left)*** *The user has drawn a few lines to indicate the shape of the skirt; the corner-detector has detected a breakpoint that the user does not want. The user will make a deletion gesture (a curve in the shape of an* α *enclosing the mistaken point) to delete it.* ***(middle)*** *The user has deleted the breakpoint, and the lines have been classified: the* silhouettes *are in red and the* borders *in yellow.* ***(right)*** *The surface inferred by the system once the user has required a reconstruction.*

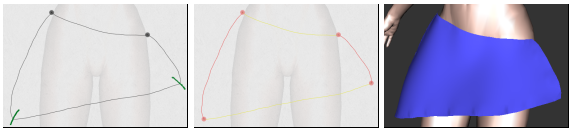

Figure 3: ***(left)*** *The user drew the outline of the skirt without sharp corners at the bottom, and the corner-detector failed to put breakpoints there. The user therefore makes gestures (overdrawn in green here) to indicate the need for new breakpoints.* ***(middle)*** *The new breakpoints have been inserted.* ***(right)*** *The skirt is reconstructed.*

the global 3D position of the cloth with respect to the body. The way they are used is detailed in Section 4.3. The two lines on the sides are classified as silhouettes, and the others are classified as border lines, as shown in the figure.

Now the user asks to see the garment inferred by the system; a surface matching the drawn constraints, but adapted to the shape of the underlying form (look near the waistine, for instance) appears almost instantly (see figure 2 (right)).

Sometimes, as in figure 3, the breakpoint-inference fails to detect all the points the user wants; in this case she or he can make a gesture to add new breakpoints.

In the current state of our system, only the front of the garment is generated; the back would have to be generated in a second pass, possibly through a simple mirror image of the silhouette strokes and user-applied modification of the border strokes. There's no texture on the surface, no indication of how a surface of this form could be constructed from flat cloth, no method for indicating that the front and back should be joined. Thus the current interface concentrates simply on the way that shape and placement can be inferred from simple strokes, not on the entire tailoring and dressing process.

2.2. Gestural interface components

The user's marks are interpreted as gestures, with the default being the construction of silhouette and border line segments. Other gestures add breakpoints for the classification process, delete breakpoints, delete a segment or an entire chain of segments, and clear all segments, as shown schematically in figure 4. (Our gestures are similar to those of Tsang et al. [TBSR04]).

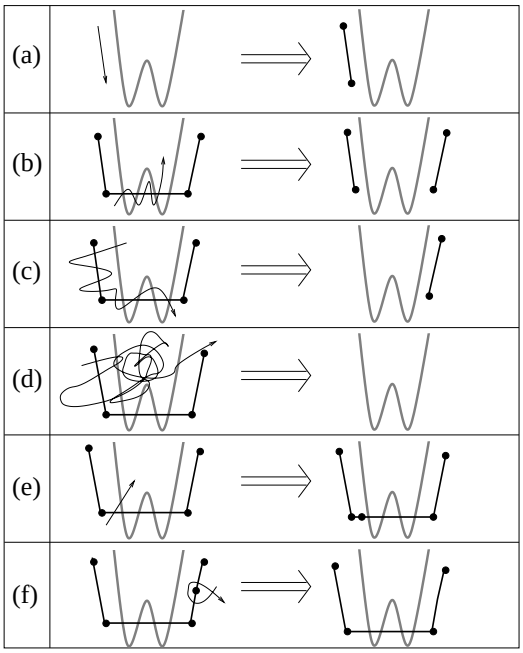

Figure 4: *The gestures in the interface; the model is drawn in thick lines, prior strokes as medium lines with dots at breakpoints, and new strokes with thin lines; the arrowhead on the new-stroke lines is not actually drawn by the user, but merely indicates the direction in which the stroke is drawn.* ***(a)*** *adding a segment;* ***(b)*** *deleting a segment (stroke must intersect the segment at least five times);* ***(c)*** *deleting several segments (the stroke must intersect more than one segment, and intersect segments at least five times. If the stroke intersects segments from two different chains, both chains are deleted entirely.);* ***(d)*** *clearing all segments (the stroke must intersect some segment and count at least fourty intersections)* ***(e)*** *adding a breakpoint* ***(f)*** *deleting a breakpoint (the stroke must intersect the segments on either side of the breakpoint, and intersect itself once).*

The breakpoint-deletion gesture matches well with the standard typsetting deletion-mark; the other deletion gestures require multiple intersections with existing strokes to prevent accidental deletions.

3. Interpreting sketches of garments: basic ideas

For the sake of clarity, we'll assume that the character is aligned with the *xy*-plane, viewed along the *z* direction.

The user draws the contours of the 2D projection of the garment in the (x,y) plane over the rendered character model. From these, we need to infer the *z*-information at every point of the contour and the interior of the garment. Clearly this problem is under-constrained; fortunately, by

considering the special nature of clothing, we can generate a plausible guess of the user's intentions. In particular, silhouettes of clothing not only indicate points where the tangent plane to the cloth contain the view direction; they usually occur as the clothing passes around the body, so we can estimate the z-depth of a silhouette edge as being the z-depth of the nearest point on the body (or, for a body placed fairly symmetrically about the xy-plane, simply use $z = 0$ as a quicker estimate). Moreover, the distance from a silhouette to the body helps us to infer the distance elsewhere, since a cloth which tightly fits the body in the (x,y) plane will also tend to fit it in the z direction, while a loose cloth will tend to float everywhere.

3.1. Overview of the algorithm

Our algorithm, whose different steps will be detailed in the following sections, develops as follows:

1. Process the 2D sketch of the garment:
   - Find high-curvature points (breakpoints) that split the contours into segments;
   - Let user add and/or delete breakpoints.
   - Classify segments of curve between breakpoints into *border lines* (which cross the character's body) or *silhouette lines*.
2. Infer the 3D position of silhouette and border lines:
   - For each breakpoint that does not lie over body, find distance to body, d, and set the point's depth, z, to the depth of the closest point on the body.
   - For each silhouette line, interpolate z linearly over the interior of the line, and use interpolated z-values to compute d-values (distance to the body in the 3D space) over the interior of the line.
   - For each border line, interpolate d over interior linearly to establish a desired distance to the model and set a z value for each point on the line;
3. Generate the interior shape of the garment:
   - Create a mesh consisting of points within the 2D simple closed curve that the user has drawn, sampled on a rectangular grid in the (x,y) plane.
   - Extend the values of d, which are known on the boundary of this grid, over the interior.
   - For each interior grid point (x,y), determine the value of z for which the distance from (x,y,z) to the body is the associated value of d.
   - Adjust this tentative assignment of z values to take into account the surface tension of the garment between two limbs of the character.
   - Tesselate the grid with triangles, clipped to the boundary curves, and render the triangles.

3.2. Pre-computing a distance field

To accelerate steps 2 and 3 of the algorithm above, a distance field to the character's model is pre-computed when the model is loaded: for each point of a 3D grid around the model, we determine and store the distance to the nearest point of the model, using the octree-based algorithm in [JS01].

The distance field will be used each time we need to find the z coordinate to assign to a point $p(x_0,y_0)$ so that it lies at a given distance from the model. This can easily be done by stepping along the ray $R(z) = (x_0,y_0,z)$ and stopping when the adequate distance value is reached (we interpolate trilinearly to estimate distances for non-grid points). When this computation is performed during a sweeping procedure, the stepping starts at the z value already found at a neighbouring pixel, which ensures the spatial coherence of the result and efficiency. Else, the process starts near the near plane of our rectangular frustum, in which the distance field has been computed.

The quality of the results depends directly on the resolution of the 3D grid storing the distance field, as does computation time. The size of the 3D grid is user-configurable, but we have generally used a $32 \times 32 \times 32$ grid.

4. Processing of contour lines

4.1. 2D processing

First, one must classify the parts of the user's sketch. As the user sketches, a new line that starts or ends near (within a few pixels of) an endpoint of an already-sketched line is assumed to connect to it. When the sketch is complete (i.e., forms a simple closed curve in the plane), we classify the parts:

- First the sketched lines are broken into segments by detecting points of high (2D) curvature (breakpoints) (this is actually done on-the-fly, as the user draws the strokes).
- Each segment is classified as a silhouette or border line: border lines are ones whose projection meets the projection of the body in the xy-plane, silhouettes are the others. The mask of the body's projection is pre-computed and stored in a buffer called the *body mask*, in order to make this classification efficient.
- The resulting segmentation is visible to the user, who may choose if necessary to add or delete breakpoints indicating segment boundaries, after which segments are reclassified.

4.2. Distance and z-value at breakpoints

Breakpoints that are located over the body model (which is tested using the body mask) are used to indicate regions of the cloth that fit very tightly to the body. They are thus assigned a zero distance to the model, and their z value is set to the body's z at this specific (x,y) location.

To assign a distance value d to a breakpoint that does not lie over the body, we step along the ray from the eye in the direction of the breakpoint's projection into the xy-plane, checking distance values in the distance field data structure as we go, and find the z-value at which this distance is minimized. We assign to the breakpoint the discovered z and d values, thus positioning the breakpoint in 3D.

4.3. Line positioning in 3D

Our aim is to use the 3D position of the breakpoints we just computed to roughly position the garment in 3D, while the garment's shape will mostly be inferred from distances to the body along the sketch silhouettes.

To position the silhouette lines in 3D, we interpolate z linearly along the edge between the two breakpoints at the extremities of the silhouette. We then set the d-values for interior points of the silhouette to those stored in the pre-computed distance field. (We could instead interpolate d directly, and compute associated z-values, but if the body curves away from the silhouette curve, there may be no z value for which the interpolated d-value is the distance to the body; alternatively, we could compute d directly for each interior point and then assign the z-value of the closest body point, as we did with the breakpoints, but in practice this leads to wiggly lines because of the coarse grid on which we pre-compute the approximate distance-to-body.)

Having established the values of z and d along silhouette edges, we need to extend this assignment to the border lines. We do this in the simplest possible way: we assign d linearly along each border line. Thus, for example, in the skirt shown above, the d-values at the two ends of the waistline are both small, so the d-value for the entire waistline will be small, while the d-values for the ends of the hemline are quite large, so the values along the remainder of the hemline will be large too.

5. 3D Reconstruction of the garment's surface

5.1. Using distance to guess surface position

As for the contour lines, the interpolation of distances to the body will be our main clue for inferring the 3D position of the interior of the garment. The first process thus consists into propagating distance values inside the garment. This is done in the following way:

The 2D closed contour lines of the garment are used to generate a (x,y) buffer (sized to the bounding box of the sketch) in which each pixel is assigned one of the values 'in', 'out' or 'border', according to its position with respect to the contour. A border pixel is one for which some contour line intersects either the vertical or horizontal segment of length one passing through the pixel-center (see figure 5); other pixels are inside or outside, which we determine using

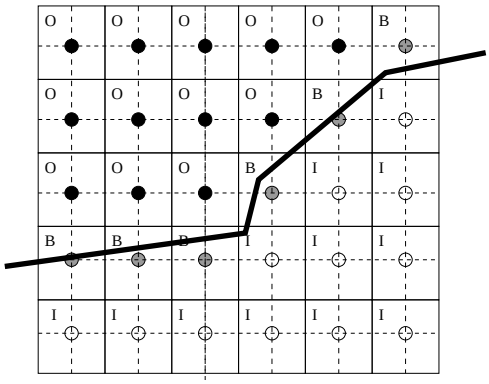

Figure 5: *Each grid point represents a small square (solid lines). If a contour (heavy line) meets a small vertical or horizontal line through the pixel center (dashed lines), the pixel is classified as a boundary; boundary pixels are marked with a "B" in this figure. Others are classified as inside ("I") or outside ("O") by winding number.*

a propagation of 'out' values from the outer limits of the buffer.

The distance values already computed along the silhouette and border lines are assigned to the border pixels. The distances for the 'in' pixels are initialized to the mean value of distances along the contour.

Then, distance information is propagated inside the buffer by applying several iterations of a standard smoothing filter, which successively recomputes each value as an evenly weighted sum (with coefficients summing to 1) of its current value and those of its "in" or "boundary" orthogonal or diagonal neighbours (see figure 6). The iterations stop when the maximum difference between values obtained after two successive iterations is under a threshold, or if a maximum number of steps has been reached. Convergence generally seems to be rapid, but we have not proven convergence of the method in general.

We then sweep in the 2D grid for computing z values at the inner pixels: as explained in section 3.2, the z value is computed by stepping along a ray in the z direction, starting at the z value we already assigned for a neighbouring point, and taking the z the closest to the desired distance value, as stored in the pre-computed distance field.

5.2. Mimicking the cloth's tension

The previous computation gives a first guess of the garment's 3D position, but still results in artefacts in regions located between two limbs of the character: due to surface tension, a cloth should not tightly fit the limbs in the in-between region (as in figure 7 (top)), but rather smoothly interpolate the limb's larges z value, due to its surface tension.

To achieve this, we first erode the 2D body mask (already mentioned in section 4.1) of a proportion that increases with

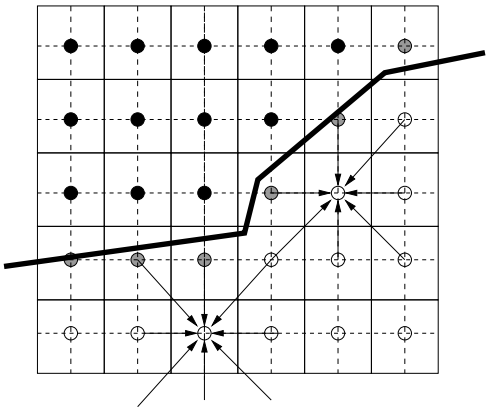

Figure 6: *The arrows indicate which neighboring pixels contribute to a pixel during the smoothing process: only neighboring* boundary *or* inside *pixels contribute to the average.*

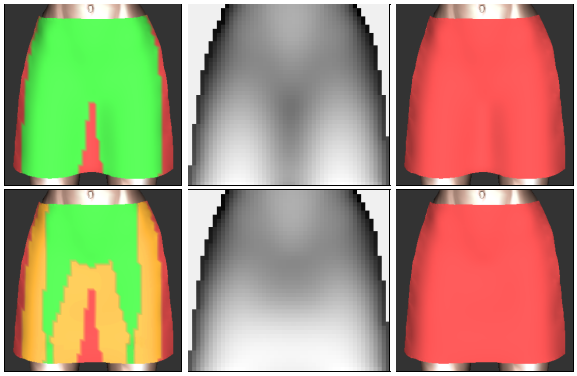

Figure 7: *Surface reconstruction without* ***(top)*** *versus with* ***(bottom)*** *taking tension effects into account. The part of the surface over the body mask is shown in green in the left images. At bottom left, the body mask has been eroded and Bézier curves used to infer the z-values between the legs. The resulting z-buffers are shown in the middle images and the reconstructed surfaces on the right.*

the underlying d value (see figure 7) (bottom left)). We then use a series of Bezier curves in horizontal planes to interpolate the z values for the pixels in-between. We chose horizontal gaps because of the structure of the human body: for an upright human (or most other mammals), gaps between portions of the body are more likely to be bounded by body on the left and right than to be bounded above and below.

To maintain the smoothness of the garment surface near the re-computed region, distances values are extracted from the new z values and the distance field. Some distance propagation iterations are performed again in 2D, before recomputing the z values int the regions not lying over the body that haven't been filled with the Bezier curves, as was done previously (see figure 7 (right)).

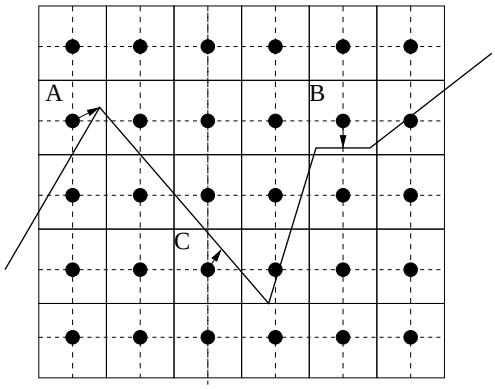

Figure 8: *When a segment ends in a pixel's box, the pixel center gets moved to that endpoint as in the box labelled "A." If more than one segment ends, the pixel center is moved to the average, as in "B". Otherwise, the pixel center is moved to the average of the intersections of segments with vertical and horizontal mid-lines, as in "C".*

We finally add a smoothing step on the z values in order to get a smoother shape for the parts of the cloth that float far from the character's model. This is done computing a smoothed version of the z-buffer from the application of a standard smoothing filter, and then by taking a weighed average, at each pixel, of the old and smoothed z-values, the weighing coefficients depending on the distance to the model at this pixel.

5.3. Mesh generation

Finally, the triangles of the standard diagonally-subdivided mesh are used as the basis for the mesh that we render (see figure 8): all "inside" vertices are retained; all outside vertices and triangles containing them are removed, and "boundary" vertices are moved to new locations via a simple rule:

- If any segments end within the unit box around the vertex, the vertex is moved to the average of the endpoints of those segments. (Because segments tend to be long, it's rare to have more than one endpoint in a box).
- Otherwise, some segment(s) must intersect the vertical and/or horizontal mid-lines of the box; the vertex is moved to the average of all such intersections.

6. Results

The examples presented in figure 1 and figure 9 were drawn in no more than 5 minutes each. We want to point out that the jagged appearance of the strokes in the drawings is simply due to the use of a mouse instead of a more adequat graphics tablet as the input device. The gallery includes simple clothes such as pullovers, skirts, robes and shirts, but also less standard ones such as an "arabian like" pair of trousers

Figure 9: *Gallery of examples of garments created with our system.*

along with basic jewelry, or exantric outfits that would certainly fit nicely in a *Haute couture* collection. This wide range of types of cloth gives an idea of the expressivity our system provides the user with.

7. Discussion

We have presented a method for dressing virtual characters from 2D sketches of their garments. We used the distance from the 2D garment silhouette to the character model to infer the variations of the distance between the garment and the character in 3D. The garment surface is generated from the silhouette and border lines and this varying distance information, thanks to a data-structure which stores the distance field to the character's body. This method has been integrated in an interactive system in which the user can interactively sketch a garment and get its 3D geometry in a few minutes.

There are other approaches that could be used as well: one could imagine a parametric template for shirts, for instance, and a program that allowed the user to place the template over a particular character and then adjust the neckline, the overall size of the shirt, etc. But this limits the realm of designs to those that exist in the library of pre-defined templates (how could one design an off-one-shoulder shirt if it wasn't already in the library), and limits it to standard forms as well: dressing the characters in a movie like *Toy Story* becomes impossible, since many of them are not human forms. Nonetheless, the model-based approach is a very reasonable one for many applications, such as a "virtual Barbie Doll," for instance.

The automated inference of the shape is simple and easy-to-understand, but may not be ideal in all cases, and we have not yet provided a way for the user to edit the solution to make it better match the idea that she or he had when sketching.

8. Future work

To infer the shape of a full garment, we could use symmetry constraints around each limb to infer silhouettes for the invisible parts, and ask the user to sketch the border lines for the back view, for instance.

We'd also like to allow the user to draw folds, and take them into account when reconstructing the 3D geometry, so that even for a closely-fitting garment, there can be extra material, as in a pleated skirt.

Most cloth is only slightly deformable, so the garment we sketch should be locally developable into a plane everywhere. We did not take this kind of constraint into account in our model, but this would be an interesting basis for future work, including the automatic inference of darts and gussets as in [MHS99].

Finally, we have sketched clothing as though it were simply a stiff polygonal material unaffected by gravity. We would like to allow the user to draw clothing, indicate something about the stiffness of the material, and see how that material would drape over the body. The difference between silk (almost no stiffness), canvas (stiff), and tulle (very stiff) can generate very different draping behaviors. In the much longer term, we'd like to incorporate a simulator that can simulate the difference between bias-cut cloth and straight-grain, the former being far more "clingy" than the latter.

References

[BCD01] BOURGUIGNON D., CANI M.-P., DRETTAKIS G.: Drawing for illustration and annotation in 3D. *Computer Graphics Forum 20*, 3 (2001). ISSN 1067-7055. 2

[BGR02] BONTE T., GALIMBERTI A., RIZZI C.: *A 3D graphic environment for garments design*. Kluwer Academic Publishers, 2002, pp. 137–150. 2

[EyHBE97] EGGLI L., YAO HSU C., BRUDERLIN B. D., ELBER G.: Inferring 3d models from freehand sketches and constraints. *Computer-Aided Design 29*, 2 (1997), 101–112. 2

[FBSS04] FLEISCH T., BRUNETTI G., SANTOS P., STORK A.: Stroke-input methods for immersive styling environments. In *Proceedings of the 2004 International Conference on Shape Modeling and Applications, June 7–9, Genova, Italy* (2004), pp. 275–283. 2

[HM90] HINDS B., MCCARTNEY J.: Interactive garment design. *The Visual Computer 6* (1990), 53–61. 2

[IH02] IGARASHI T., HUGHES J. F.: Clothing manipulation. In *Proceedings of the 15th annual ACM symposium on User interface software and technology* (2002), ACM Press, pp. 91–100. 2

[IMT99] IGARASHI T., MATSUOKA S., TANAKA H.: Teddy: a sketching interface for 3d freeform design. In *Proceedings of the 26th annual conference on Computer graphics and interactive techniques* (1999), ACM Press/Addison-Wesley Publishing Co., pp. 409–416. 2

[JS01] JONES M. W., SATHERLEY R.: Using distance fields for object representation and rendering. In *Proceedings of the19th Ann. Conf. of Eurographics (UK Chapter)* (2001), pp. 91–100. 4

[MHS99] MCCARTNEY J., HINDS B. K., SEOW B. L.: The flattening of triangulated surfaces incorporating darts and gussets. *Computer Aided Design 31*, 4 (1999), 249–260. 8

[TBSR04] TSANG S., BALAKRISHNAN R., SINGH K., RANJAN A.: A suggestive interface for image guided 3d sketching. In *Proceedings of CHI 2004, April 24–29, Vienna, Austria* (2004), pp. 591–598. 3

[WMTT93] WERNER H. M., MAGNENAT-THALMANN N., THALMANN D.: User interface for fashion design. In *Proceedings of the IFIP TC5/WG5.2/WG5.10 CSI International Conference on Computer Graphics* (1993), North-Holland, pp. 197–204. 2

[ZHH96] ZELEZNIK R. C., HERNDON K., HUGHES J. F.: Sketch: An interface for sketching 3d scenes. In *Computer Graphics Proceedings, SIGGRAPH'96* (New Orleans, Louisiana, August 1996), Annual Conference Series, ACM. 2