Interactive Assembly and Animation of 3D Digital Garments

O. Nylén[1†] P. Pall[1†] Y. Ishiwaka[2] K. Suda[2] M. Fratarcangeli[3]

[1]Deform Dynamics [2]SoftBank Corp. [3]Chalmers University of Technology

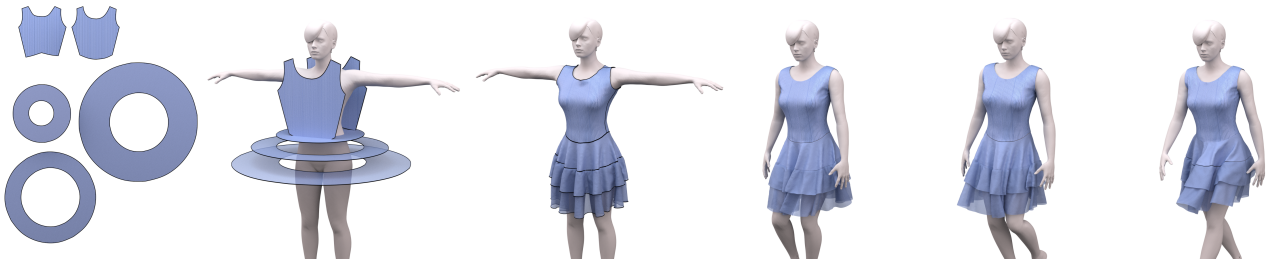


Figure 1: *By using our tool, multiple layers of 2D patterns are placed around a virtual character and stitched together interactively.*

Abstract

We present a novel real-time tool for sewing together 2D patterns, enabling quick assembly of visually plausible, interactively animated garments for virtual characters. The process is assisted by ad-hoc visual hints and allows designers to import 2D patterns from any CAD-tool, connect them using seams around a 3D character with any body type, and assess the overall quality during the character animation. The cloth is numerically simulated including robust modeling of contact of the cloth with itself and with the character's body. Overall, our tool allows for fast prototyping of virtual garments, achieving immediate feedback on their behaviour and visual quality on an animated character, in effect speeding up the content production pipeline for visual effects applications involving clothed characters.

CCS Concepts

• ***Computing methodologies*** → ***Shape modeling; Mesh models;*** *Collision detection;*

1. Introduction

The manufacturing of clothes in the physical world relies heavily on the use of two-dimensional patterns. These are templates capturing the dimensions and the shape of a specific garment body [Fas98]. The design of virtual garments in the apparel industry and entertainment (games, movies) relies on similar mechanisms. We present a novel system for semi-automatically sewing the contours of triangulated patterns together, to create digital garments usable in real-time design and animation. The system provides an user interface for creating seams between the patterns in 3D. The cloth, including the seams, is modeled as a nonlinear constrained particle system according to *Projective Dynamics* [BML*14], and solved on the GPU in real-time using the *Vivace* solver [FTP16]. Overall, our system enables visual effects artists to quickly assemble virtual garments for animated characters in real-time with performance and visual quality suitable for performance-driven animation (Figures 1,6).

2. Related Work

One of the open challenges in computer graphics and engineering is achieving interactive, physics-in-the-loop design solutions for practically relevant problems.

Cloth simulation. In the last decades, the dynamics of cloth have been formulated according to different mechanical theories, including mass-spring systems and continuum-based systems (e.g., see [MBT*12]). While accurate methods are able to realistically capture most of the cloth dynamics, they are computationally expensive. Recently, novel methods have addressed the gap between accuracy and performance by mapping high-performance nonlinear models [MMCK14, BML*14] to the massively parallel GPU [Wan15, FTP16, PNF18, WWF*18].

† Shared first authorship. These authors contributed equally to this work.

DOI: 10.2312/egs.20201021

Garment and pattern design. In the context of garment design, several state-of-the art academic frameworks have been presented. A representative sample is *sensitive couture* [UKIG11], which provides a continuous, natural design modality of 2D patterns that has also been expanded with machine learning to automatically parse sewing patterns [BGK*13]. Their system allows to seamlessly transfer edits to the corresponding 3D cloth model on a virtual character, but it does not support self-collisions of the cloth which makes it difficult to accurately preview the results, in particular when the character is animated. Other commercial frameworks include *CLO3D* and *Tuka3D*, which are widely used in the industry even though they have limited support for real-time character animation.

3. System Overview

The physical 2D patterns used for clothes manufacturing do not only express the shape of the garment patches, but also detail how pieces of fabric are cut and sewn together. Popular file-formats for digitally storing these patterns include CAD formats such as DXF or plain PDF files. In some cases, these files include explicit information about how seams have to be created, but in general the process of creating garments based on a sewing pattern is a task that requires human expertise. Our system assists the designer in semi-automatically sewing the contours of triangular meshes together, to create digital garments and use them in 3D real-time simulation. The system provides a user interface for creating seams between objects in 3D (Sec. 4). The properties of the cloth can then be set up using an *ad-hoc* painting tool (Sec. 5), and finally the garment is assembled and animated including collision handling (Sec. 6).

4. Sewing System

In this section, we provide an overview of the user-interface functionalities to stitch virtual patterns together.

Manual seam definition. Seams are defined as a one-to-one mapping between two ordered sets of n vertices each. The two sets of vertices belong to the external contours C_A and C_B of two triangulated patterns A and B respectively (Fig. 2 *left*). The workflow to define the seams is divided into the following steps:

1. A vertex a_1 belonging to the contour C_A is selected as the first endpoint of the seam on A.
2. By hovering the mouse, the shortest path from a_1 to the vertex currently nearest to the mouse cursor lying on C_A is highlighted;
3. By clicking, the selected vertex a_n, is picked as the other endpoint of the seam. This implicitly defines the number of vertices (n) in the shortest path along C_A between a_1 and a_n;
4. A vertex b_1 belonging to the contour C_B is selected as the first endpoint of the seam on B;
5. By hovering the mouse, the system highlights the shortest path composed by n vertices, lying on C_B and starting from b_1. There are two potential options for selecting this path, going clockwise or counter-clockwise from b_1. Only one option is highlighted at a time, and which one to highlight is determined by computing the shortest distance between the mouse cursor and the two potential endpoints b_n. By clicking, b_n is defined as the last point of the currently highlighted path.

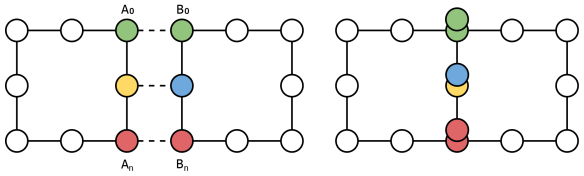


Figure 2: *Left. A seam is defined between two patches. Right. Distance constraints with zero rest length bring the particles together.*

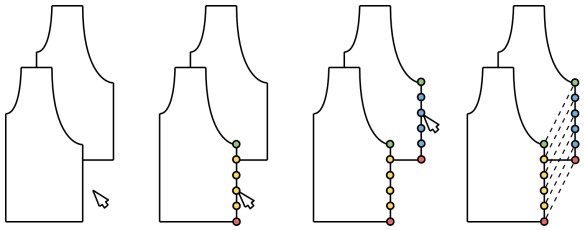
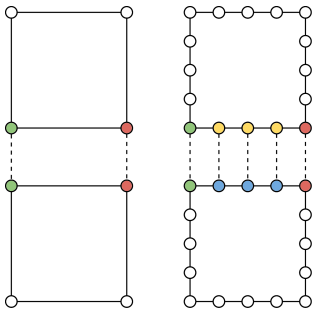


Figure 3: *Creating a seam with automatic suggestions.*

Automatic seam suggestion. Often, sewing is defined between two symmetric portions of contours delimited by corners, that is where the angle ϕ between two neighboring edges lying on the contour is smaller than a user-defined threshold. In this case, the procedure to define a seam can be simplified as depicted in Fig. 3. The user hovers the mouse over the contour of the triangulated pattern, and the nearest portion of the contour delimited by two corners is highlighted. When the mouse button is clicked, the contour is selected. The same is repeated for the other side of the seam.



Adapting to topological change. It might occur that a user wants to change the polygonal resolution (hence the topology) of a triangulated pattern *after* the sewing with other patterns have been defined. This is because the topology of the mesh can influence the cloth dynamics, and the user may want to change it to achieve particular effects (e.g., wrinkly materials, like silk, require more triangles than more rigid materials like cotton). When the mesh topology changes, the seam will in general not be valid. To keep the seams consistent, the system automatically redefines the seam for the new topology. By comparing the positions of a_1, a_n, b_1 and b_n in the original mesh to their counterparts in the updated mesh, their indices (and the indices of the vertices in between) are recomputed using the list of contour points for the updated mesh.

Bending two-dimensional patterns. The 2D patterns may have to be wrapped around objects, e.g., sleeves around arms. In this case, the corresponding triangular mesh needs to be bent. If the patch would be left flat, in fact, the seam lines would go through

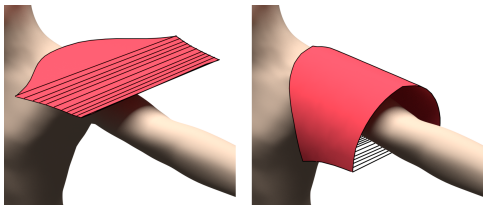


Figure 4: *A flat sleeve patch is interactively bent and wrapped around the arm.*

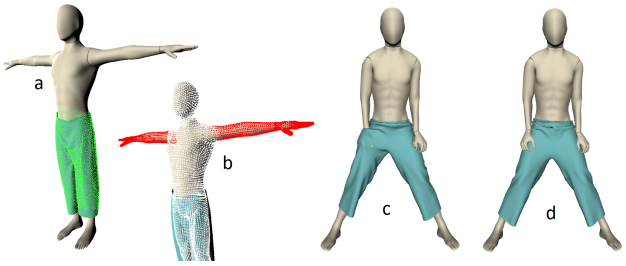


Figure 5: *Painting per-particle parameters. The collision levels of a pair of pants (green, a) and arms (red, b). By using collision levels, the undesired collisions (c) are ignored (d).*

the patch. To address this case, we bend the triangulated patterns in 3D by interactively rotating them around an axis defined by the user. The amount of rotation is scaled with the distance from the center of the pattern, ie. vertices further from the center are rotated more. Optionally, the user can also set a symmetry axis to bend the mesh symmetrically, and a distance threshold from the rotation axis for where the bending starts.

Seam simulation. The seams between corresponding vertices are modeled as distance constraints with zero rest length. In this way, as soon as the simulation starts, the vertices connected by a seam are always kept at the same position during the animation. Since seams influence the bending behavior of garments in a significant way [PKST08], we also define hinge-edge constraint by the quadratic bending energy [BWH*06] on the overlapping sewn edges. Both bending and distance stiffness can be defined by the user for each individual seam.

As the simulation starts, the initial displacement caused by the distance constraints is rather abrupt as the vertices are brought from their initial to their "stitched" position in a very short time. To avoid any instability, we increase the air damping until the position of the corresponding vertices match. Once the final position is reached for most particles (i.e., the residual error for the distance constraints is very low), the damping is linearly interpolated to its initial value. See the accompanying executable demo for a demonstration of this mechanism.

5. Painting Per-Particle Parameters.

We implemented a painting tool to assign binary, integer and floating point values to parameters of individual particles, e.g., the amount of kinetic and static friction, mass and collision levels. A collision level is useful to disable collisions in some areas of a garment and the colliders. By default, all simulated particles collide with all the colliders. By assigning a different collision level to an area of the garment and a different one to another area of the collider, however, the collision tests with each other are disabled. This is particularly useful when animating a motion-captured character. The virtual avatar may have different shape than the human actor and limbs may intersect and interpenetrate. By painting collision levels, the system is less prone to errors (Fig. 5).

6. Collision System

In this section, we briefly depict our mesh- and self-collider, used respectively for the collision of the cloth with itself, and for the collision between the cloth and the character. Collisions are tested at the end of each time step (discrete collision detection), and we employ a GPU-based implementation of a spatial hash map as data structure to store the geometric primitives [THM*03]. In this way, we reduce the amount of accesses to video memory and fetch directly the colliding primitives without traversing bounding volume hierarchies, while keeping the memory footprint contained.

Spatial hash map. Here, we briefly summarize the basic mechanism of a spatial hash map. The reader is referred to [THM*03] for further details. For each time step, the hash map is rebuilt from scratch.The hash index h is computed as:

$$h = |(i_x \times p_1) \oplus (i_y \times p_2) \oplus (i_z \times p_3) \bmod n| \tag{1}$$

where

$$i_x = \lfloor \frac{x}{l} \rfloor \qquad i_y = \lfloor \frac{y}{l} \rfloor \qquad i_z = \lfloor \frac{z}{l} \rfloor. \tag{2}$$

x, y and z are the scalar coordinates of 3D primitives, l is the length of the cell side, p_1, p_2 and p_3 are very large integer prime numbers, and n is the size of the hash map. Note that Eq. 1 is slightly different from the original formula in [THM*03] allowing to avoid the bounding box computation of the space domain. Since n is smaller than the actual number of cells in the space domain, it may happen that primitives in different cells may be stored in the same bucket. To figure out which primitives belong to the same cell without comparing their positions, we associate a unique `cell_id` computed as:

$$z \times p_1 \times p_1 + y \times p_1 + x. \tag{3}$$

The self-collider. For the collisions of the cloth with itself, we associate a sphere with each vertex and then the collisions are handled using a discrete sphere-sphere test after each solver iteration, similarly to [MMCK14]. The spheres are tested only if they are stored in the same bucket in the hash map and have the same `cell_id`. We found empirically that the best performance was obtained by setting l in the hash map as the average length of the cloth edges.

The mesh collider. In the case of the mesh collider, used for testing the collisions between the character and the cloth, each triangle of the collider is inserted in the hash map. This is done according to Alg. 1.

Algorithm 1: Building the hash map

```
for each triangle T_i do in parallel
    Calculate AABB(T_i)
    for each (cell_j ∩ AABB(T_i) ≠ ∅) do
        if ||t_k^i − cell_j|| < l/√2, k=1, 2, 3 then
            atomic insert i at h(cell_j)
    end
end
```

In step 1-2, the axis-aligned bounding box of each triangle T_i is computed. Then, each cell that is intersected by $AABB(T_i)$ is iterated over in step 3. In step 4-5, if one of the triangle vertices $t_k^i, k = 1,2,3$ is nearer than $l/\sqrt{2}$ to the center of the cell, then we insert the triangle index i in the cell.

In the collision handling, each cloth particle is processed in parallel on the GPU. The particle is checked against each triangle in the cell with the same `cell_id`. If the particle is found to be on the opposite side of all triangles in the cell w.r.t. the triangle normal, a collision has been found and we define a temporary *inequality* constraint to move the particle to the closest point on the surface of the nearest triangle. The inequality constraints are removed at the end of the time step. For friction, we use the Coulomb model as in [MMCK14], considering the relative velocities between the triangle and the colliding particles. To reduce the effect of *tunneling* (an inherent problem with discrete collision detection), we linearly interpolate the triangles of the collider mesh each time step instead of simply using the input triangles which are updated once during each frame.

When using a uniform data structure like the spatial hash map, the performance is improved by using isotropic triangles with approximately the same size. In this way, each cell stores approximately the same number of triangles. This condition, however, is not met by most of the character meshes. Therefore, we provide the user with the possibility to remesh the collider mesh, which is different from the input mesh used for rendering, using [LWL*09].

7. Discussion and Limitations

Our system is particularly suitable for virtual productions where massive amounts of content must be created quickly, like modern real-time film-making relying on performance-driven animation (which is currently taking over the visual effects industry). Our system is fast, robust, and plausible enough to produce believable cloth animations on different body types, allowing rapid prototyping and fast iterations (*see accompanying video and demo*). One considerable limitation is the fact that the current sewing system requires the number of vertices on both sides of the seam to be equal. Our future research will focus on generalizing the stitching functionalities, and implementing more accurate friction models, continuous collision detection and precise nonlinear solvers.

Figure 6: *2D triangulated patterns are stitched around a virtual character. See the accompanying material for the animations.*

References

[BGK*13] BERTHOUZOZ F., GARG A., KAUFMAN D. M., GRINSPUN E., AGRAWALA M.: Parsing sewing patterns into 3d garments. *ACM Trans. Graph. 32*, 4 (July 2013), 85:1–85:12. 2

[BML*14] BOUAZIZ S., MARTIN S., LIU T., KAVAN L., PAULY M.: Projective dynamics: Fusing constraint projections for fast simulation. *ACM Trans. Graph. 33*, 4 (July 2014), 154:1–154:11. 1

[BWH*06] BERGOU M., WARDETZKY M., HARMON D., ZORIN D., GRINSPUN E.: A quadratic bending model for inextensible surfaces. In *Eurographics Symposium on Geometry Processing* (2006), SGP '06, pp. 227–230. 3

[Fas98] FASANELLA K.: *The Entrepreneur's Guide to Sewn Product Manufacturing*. Apparel Technical Svcs, 1998. 1

[FTP16] FRATARCANGELI M., TIBALDO V., PELLACINI F.: Vivace: A practical gauss-seidel method for stable soft body dynamics. *ACM Trans. Graph. (Siggraph ASIA) 35*, 6 (Nov. 2016), 214:1–214:9. 1

[LWL*09] LIU Y., WANG W., LÃL'VY B., SUN F., YAN D. M., LU L., YANG C.: On centroidal voronoi tessellation - energy smoothness and fast compu tation. *ACM Transactions on Graphics* (2009). 4

[MBT*12] MIGUEL E., BRADLEY D., THOMASZEWSKI B., BICKEL B., MATUSIK W., OTADUY M. A., MARSCHNER S.: Data-driven estimation of cloth simulation models. *Comput. Graph. Forum 31*, 2pt2 (May 2012), 519–528. 1

[MMCK14] MACKLIN M., MÜLLER M., CHENTANEZ N., KIM T.-Y.: Unified particle physics for real-time applications. *ACM Trans. Graph. 33*, 4 (July 2014), 153:1–153:12. 1, 3, 4

[PKST08] PABST S., KRZYWINSKI S., SCHENK A., THOMASZEWSKI B.: Seams and Bending in Cloth Simulation. In *Virtual Reality Interactions and Physical Simulation, VRIPHYS* (2008). 3

[PNF18] PALL P., NYLÉN O., FRATARCANGELI M.: Fast Quadrangular Mass-Spring Systems using Red-Black Ordering. In *Virtual Reality Interaction and Physical Simulation, VRIPHYS* (2018). 1

[THM*03] TESCHNER M., HEIDELBERGER B., MÃIJLLER M., POMERANETS D., GROSS M.: Optimized spatial hashing for collision detection of deformable objects. *Vision, Modeling, Visualization, VMV 3* (12 2003). 3

[UKIG11] UMETANI N., KAUFMAN D. M., IGARASHI T., GRINSPUN E.: Sensitive couture for interactive garment editing and modeling. *ACM Transactions on Graphics (SIGGRAPH 2011) 30*, 4 (2011). 2

[Wan15] WANG H.: A chebyshev semi-iterative approach for accelerating projective and position-based dynamics. *ACM Trans. Graph. 34*, 6 (Oct. 2015), 246:1–246:9. 1

[WWF*18] WANG Z., WU L., FRATARCANGELI M., TANG M., WANG H.: Parallel Multigrid for Nonlinear Cloth Simulation. *Computer Graphics Forum* (2018). 1