Deferred Warping

Martin Knuth, Jan Bender, Michael Goesele, Arjan Kuijper

Abstract—We introduce *deferred warping*, a novel approach for real-time deformation of 3D objects attached to an animated or manipulated surface. Our target application is virtual prototyping of garments where 2D pattern modeling is combined with 3D garment simulation which allows an immediate validation of the design. The technique works in two steps: First, the surface deformation of the target object is determined and the resulting transformation field is stored as a matrix texture. Then the matrix texture is used as look-up table to transform a given geometry onto a deformed surface. Splitting the process in two steps yields a large flexibility since different attachment types can be realized by simply defining specific mapping functions. Our technique can directly handle complex topology changes within the surface. We demonstrate a fast implementation in the vertex shading stage allowing the use of highly decorated surfaces with millions of triangles in real-time.

Index Terms—real-time deformation, garment modeling, virtual prototyping

1 Introduction

For many years the simulation of cloth models has played a significant role in computer games and movies. In the last years it also became important in virtual garment prototyping, which is an essential part of the garment manufacturing pipeline. In a virtual prototyping process a designer models a 2D garment pattern consisting of multiple pieces of cloth, which are sewed together in a 3D cloth simulation. The simulation allows the designer to validate the garment design and to guarantee producibility. Current real-time cloth simulation methods can often only handle coarse meshes. These meshes are textured in the rendering process to obtain more realistic results. However, the area of virtual garment prototyping demands more detailed models since coarse textured meshes often do not look realistic in a close-up view, e.g., when representing a knitwear model. Furthermore, there exist a set of typical objects (appliqué), which are attached to the garment surface as decoration (e.g., sequins) or which have a functional task (e.g., buttons). Simulating knitwear models or garments with many attached objects is computationally too expensive for a real-time simulation in an interactive virtual prototyping system.

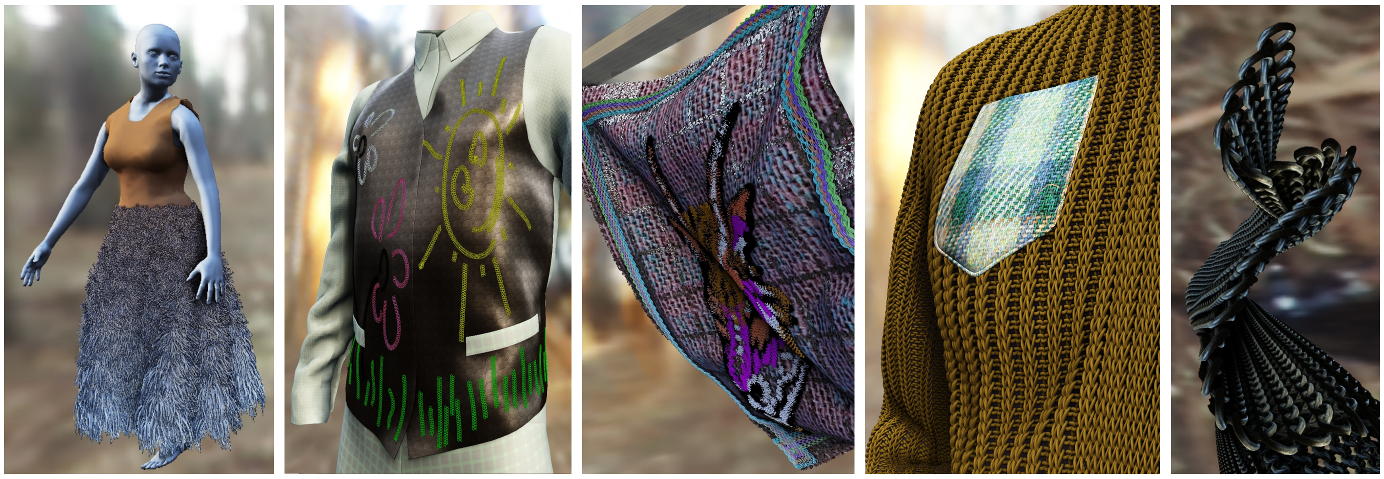
In this paper we introduce *deferred warping*, which allows the attachment of arbitrary geometry on deformed garment models. The key idea of our approach is to enrich coarse simulation models with geometric details in a post-processing step. Our method allows a high quality visualization of simulated garment models with many geometric details like appliqué or even the replacement of the cloth surface with a complex knitwear model at interactive frame rates.

- *Martin Knuth - Fraunhofer IGD*
  E-mail: martin.knuth@igd.fraunhofer.de
- *Jan Bender - Graduate School CE, TU Darmstadt*
  E-mail: bender@gsc.tu-darmstadt.de
- *Michael Goesele - TU Darmstadt*
  E-mail: michael.goesele@gris.informatik.tu-darmstadt.de
- *Arjan Kuijper - Fraunhofer IGD*
  E-mail: arjan.kuijper@igd.fraunhofer.de

Appliqué differ strongly in the way they are attached to a garment surface, e.g., buttons are fixed at a single point, seams are attached along a curve, and patches are fixed on a particular area of the surface (see also the examples in Figure 1). Our method is able to handle these different attachment types while using UV coordinates to define the positions of the detail geometries on the deformed surface. The area attachment can even be used to replace the complete surface with a detailed geometric cloth or a knitwear model as shown in Figure 1. This gives us the flexibility required for interactive virtual garment design.

Our deferred warping approach consists of two steps: First, we create an intermediate representation of the surface deformation stored as a matrix texture. The matrix texture defines the position in 3D world coordinates and the corresponding local tangent space orientation for each vertex. This allows us to define and interpolate the deformation for any surface point while deferring the actual warp to the second step. The deformation field has to be extracted only once and potentially only at coarse locations yielding a significant performance gain. In the second step we use this information to deform the attached geometry. This allows the flexible attachment of geometric details in various ways. The whole process is independent of the target surface's topology and resolution as long as a suitable UV parametrization exists. Note that the definition of unique and orthogonal UV coordinates on a 2D garment pattern is trivial. Each piece of cloth of a garment pattern has its own material and UV parametrization. This results in visible seams when the pieces are sewn together. This is desired in virtual garment prototyping since the seams are an important tool for the visual design of a garment [1].

Deferred warping was originally developed for garment simulations but it can be used in combination with any animation, manipulation, or simulation system. Geometric details can be added without increasing the computational effort of the animation system itself. The proposed method can be easily implemented in the vertex shading stage of current GPU rendering pipelines and combined with other techniques. E.g., we can still use geometry instancing to



(a) (b) (c) (d) (e)

Fig. 1. Deferred warping allows to attach arbitrary detail geometry to an animated or manipulated surface in real-time as required in interactive garment simulations. Details can be attached at single points (a,c), along a curve (b,c), or on an area (d,e). In the latter case, the original geometry was even completely replaced with geometry representing the fabric structure.

attach many geometrical details to an existing surface. Integrating the algorithm into a renderer does not significantly increase the amount of data needed for rendering or the computational complexity. It is therefore well-suited for interactive applications.

Our contributions are:

- A technique to attach detail geometry to an animated or deforming target surface that separates the computation of the transformation from its application to the detail geometry,
- several different ways to attach and deform detail geometry, including point, curve, and area attachments, and
- an efficient implementation in the vertex processing stage of modern GPUs with real-time performance and seamless integration with other rendering techniques.

2 Related Work

In the last years, the increasing performance of computer hardware and the development of efficient cloth simulation methods have established garment simulation in design tools for the garment and fashion industry [2]. However, state-of-the-art simulation methods are still not able to handle garments with a large numbers of appliqué in real-time.

In computer graphics there exist a wide range of approaches to attach details to a smooth surface and to visualize them. But none of those is flexible enough to attach geometric details in various ways to a deforming, potentially interactively manipulated, or even remeshed surface while still yielding interactive frame rates. This is, however, required in virtual garment prototyping.

Decal rendering is a mapping method in which a texture patch is projected locally onto a geometry. The coordinate computation can be difficult in this case. A solution for this problem is presented by Schmidt et al. [3], which computes the needed coordinates on-the-fly using exponential maps. Zhou et al. [4] focus on the structure of the underlying parametrization of the base mesh in order to increase the quality of the attached geometry. To visualize the attached geometry they use a ray casting approach. The generation of the underlying parametrization is also a topic of the work of Ma et al. [5] who present a technique to fill a user-defined volume with aligned 3D geometry.

For area texturing with a repeating detail there exist several methods to apply displacements to a surface. Height-field based techniques allow to handle self-occlusion using a ray casting step which uses binary search to find the nearest intersection with the viewing ray. This so-called *relief mapping* can be extended to multiple layers in order to represent 3D objects within a surface as shown by Policarpo and Oliveira [6]. Neyret [7] uses volumetric textures instead of layers to represent the detail geometry, which allows a fast computation and the use of complex detail geometry.

Toledo et al. [8] present a technique, that uses patches with overlapping height field information in order to represent complete highly detailed 3D models. The technique is capable of representing highly detailed static geometry. Changes of the surface topology or animation require a recomputation of the height fields, which represent characteristic parts of the surface.

Wang et al. [9] introduce a real-time approach based on five-dimensional *Generalized Displacement Maps* (GDM). These maps are used to speed up a ray marching process that creates the image of the detail geometry. To perform the ray marching, the target surface is represented by prisms, which are used to map rays from world to local surface space. Porumbescu et al. [10] use a very similar approach. Instead of using a 5D-map, the detail geometry is traced directly. They warp geometric detail inside a tetrahedral cage layer attached to the target surface also known as Shell Mapping. Jeschke et al. [11] improve on this by ensuring a consistent warping between neighboring cages. Shen [12] extends Shell Mapping to support the animation of a shell, which allows morphing and deformation of the surface objects. Ritsche [13] presents a real-time version of a shell space rendering approach. The technique uses ray tracing of surface details stored as volumetric textures. The shell space is precomputed and thus static.

Brodersen et al. [14] later extended this for implicit representations, allowing advanced attachment modes such as sum and subtraction. This can be used to create a smooth transition between target surface and detail geometry. They also propose a near real-time variant where the detail geometry is represented explicitly.

A method for the simulation and rendering of knitted garments is presented by Yuksel et al. [15]. Their approach uses two stages. The first stage operates on a quad based mesh and is used for overall relaxation. These quads are filled with stitch meshes, which represent yarn based geometry. Their approach solves the mapping problem by using quads for the base mesh. The mapping process is, however, intended for offline rendering.

An approach similar to ours, but limited to area mapping, is introduced by Schein et al. [16]. The authors present a real-time implementation of *Deformation Displacement Mapping*, which was originally introduced by Elber et al. [17]. In order to map geometry onto a target surface they compute a position and normal texture from the surface. This step is performed as an offline process while the deformation mapping itself is performed in real-time on the GPU. Due to the offline process animated surfaces are not directly supported. However, today this preparation step can be performed on current graphics hardware.

The purpose of all techniques described above is to handle decal and area mappings. It is difficult to extend them to handle curve-based features, such as embroidery, which are important in virtual garment prototyping. Cage-based deformation methods are able to handle such features. We note that the prism-based approaches could be used as a cage-based deformation system for detail geometry. However, such an approach would need a very smooth interpolation function. These are typically considered as the holy grail of cage-based deformation. Interpolation functions such as high order barycentric coordinates [18] or Green coordinates [19] would be required to create smooth transitions. These functions are, however, very complex, which would reduce the performance by a large degree compared to prisms with hard boundaries.

In conclusion we can state three main differences of our work to the existing approaches:

First, the presented methods with the exception of Schein et al. [16] use a representation that requires raytracing techniques to render the detail geometry or approximated data. In contrast, our method as well as the one of Schein et al. [16] simply render the geometry in the same way the target geometry is rendered using arbitrary rendering systems including rasterization. Both methods can also post-process the attached geometry within the rendering pipeline as needed. We show an example with further deformation later on in Section 3.6.

Second, in garment simulation the patterns have a perfect UV parametrization per definition. However, the simulation can shear the parametrization until it degenerates. Shell space and previous geometry based methods are intended to be used as texturing methods for 3D meso-structures and are dependent on the UV parametrization of the base mesh. This means that they are also affected by shearing of the texture coordinate space, which is a large problem for our target application of garment design (e.g., buttons must not be sheared). Therefore, in contrast to all other previous methods, our approach defines a complete orthogonal reference frame at every vertex position. Having a full reference frame allows us to support advanced attachment types such as point and curve attachments, independent of the status of the base surface. Note that these types do not shear the detail geometry even in case of a sheared UV parametrization.

Third, we need no offline pre-processing step since our matrix textures are computed on-the-fly. This enables us to attach and modify detail geometries on animated or manipulated surfaces. Moreover, the contained matrices can be modified in this process providing more flexibility to the user.

3 Deferred Warping

In this section we describe the deferred warping concept in detail. Our method works in two steps: First, the transformation field of a deformed surface is extracted on-the-fly. Second, this field is used to transform detail geometry on the surface. We also call this attaching geometry to a surface. Note that we use the terms *source object* for the geometry we want to attach to the deformed surface and *target object* for the externally animated or deformed geometry. We assume that a suitable parametrization of the target object exists. If this is not the case, such a parametrization can be generated using well-known techniques (see, e.g., Akenine-Möller et al. [20] or Hormann et al. [21]). Note that in our application area of virtual garment prototyping the parametrization is given by the UV coordinates of the garment pattern.

In the following we first give an overview of deferred warping. Then we show how the transformation field of a deformed surface is determined. Finally, we demonstrate how different attachment types are realized by simply defining two mapping functions and introduce the most important types (see Figure 2). These attachment types can be used to implement a large variety of effects (see Figure 1). Deferred warping is, however, not limited to these types. Other definitions are also possible, e.g., to support the animation of the source geometry on the target surface (see Figure 8).

3.1 Overview

In the first step of our method the transformation field of a deformed surface is determined, which is defined by a 3×4 linear transformation matrix for each vertex. This field defines a look-up table from texture space into tangent space. Having the transformation field, we can define a *matrix texture* as the function $\mathbf{f}_{\text{mt}} : [0,1]^2 \mapsto \mathbb{R}^{3\times4}$, which maps the surface parameter coordinates (u, v) to the corresponding 3×4 matrix (see Section 3.2). Hence, this matrix texture yields a local coordinate system on the deformed surface of the target object at any surface position (u, v).

In the second step the matrix texture is used to implement different attachment types. To obtain the corresponding local coordinate system for a point of the detail geometry, we need a mapping function $\mathbf{f}_{\text{uv}} : \mathbb{R}^3 \mapsto \mathbb{R}^3$, which

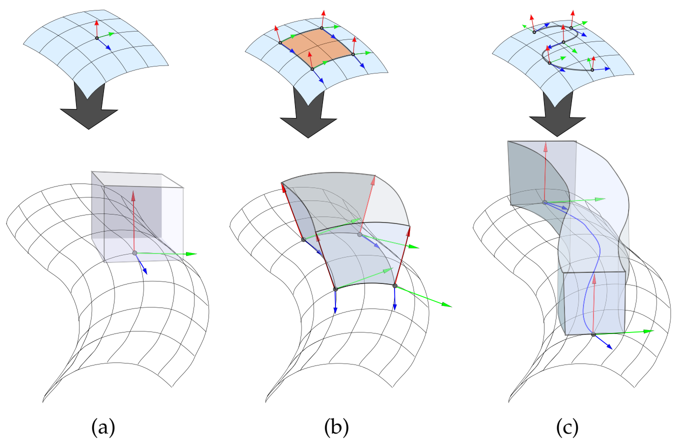


Fig. 2. Possible attachments of a 3D geometry to a 2D surface. (a) Point attachment: The geometry is directly transformed by the matrix retrieved from the transformation field for the given position, placed on the surface, and oriented according to the principal axes of the subspace. (b) Area attachment: Texture coordinates are determined by projecting the source geometry on the xz-plane. The source geometry follows the curvature of the target surface. The y-coordinate of a source point defines the distance to the target surface in normal direction. (c) Curve attachment: A user-defined parametric curve function defines the texture coordinates for the matrix look-up. One component of a source point is used as parameter for the curve while the other components determine the final position in the plane which is orthogonal to the curve.

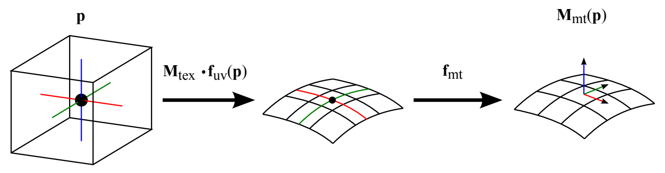

$\mathbf{p}$
$\mathbf{M}_{\text{tex}} \cdot \mathbf{f}_{\text{uv}}(\mathbf{p})$
$\mathbf{f}_{\text{mt}}$
$\mathbf{M}_{\text{mt}}(\mathbf{p})$

Fig. 3. To determine the matrix $\mathbf{M}_{\text{mt}}$ for a point $\mathbf{p}$, first a projection of $\mathbf{p}$ onto the target surface is performed. Then the tangent space is determined for the projected point which is a subspace aligned to the UV parametrization of the surface.

yields a valid homogeneous texture coordinate $(u, v, 1)$ for each detail point. This function can be defined in different ways to create different attachment types (see Sections 3.3-3.5). Furthermore, we introduce the matrix $\mathbf{M}_{\text{tex}} \in \mathbb{R}^{2\times3}$ to allow for a linear transformation in texture space. This gives the user the possibility to change the position, rotation, and scale of an attachment on the target surface easily and dynamically. Finally, the local coordinate system of a detail point $\mathbf{p}$ on the deformed surface is determined by the matrix $\mathbf{M}_{\text{mt}}(\mathbf{p})$:

$$\mathbf{M}_{\text{mt}}(\mathbf{p}) = \mathbf{f}_{\text{mt}}\left(\mathbf{M}_{\text{tex}} \cdot \mathbf{f}_{\text{uv}}(\mathbf{p})\right). \tag{1}$$

Figure 3 illustrates the computation of $\mathbf{M}_{\text{mt}}(\mathbf{p})$ for a point $\mathbf{p}$. The mapping function $\mathbf{f}_{\text{uv}}$ only depends on $\mathbf{p}$. However, deferred warping supports the use of arbitrary functions as long as they provide valid texture coordinates. For example, an animated attachment can be implemented using a mapping that depends on a time parameter.

The matrix $\mathbf{M}_{\text{mt}}$ can be used to transform detail geometry on the target surface. But instead of applying this transformation directly, we introduce another transformation matrix $\mathbf{M}_{\text{tan}} \in \mathbb{R}^{4\times4}$ to give the user more flexibility. This matrix provides the possibility to transform the detail geometry in tangent space before attaching it to the deformed surface, e.g., to scale the geometry in direction of

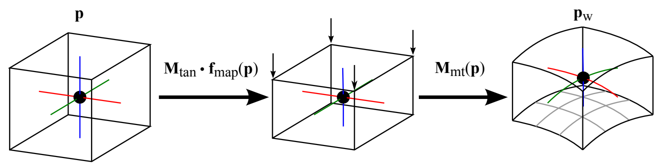

$\mathbf{p}$
$\mathbf{M}_{\text{tan}} \cdot \mathbf{f}_{\text{map}}(\mathbf{p})$
$\mathbf{M}_{\text{mt}}(\mathbf{p})$
$\mathbf{p}_{\text{w}}$

Fig. 4. To transform a detail point $\mathbf{p}$ to world space, we apply a mapping function $\mathbf{f}_{\text{map}}$ that depends on the attachment type. The resulting position is first transformed by a user defined matrix $\mathbf{M}_{\text{tan}}$, which yields more flexibility, and then by the matrix $\mathbf{M}_{\text{mt}}$ to transform the result onto the deformed surface.

the surface normal (see Figure 4, middle). The final position of an attached detail point $\mathbf{p}$ in world space is determined by:

$$\mathbf{p}_{\text{w}} = \mathbf{M}_{\text{mt}}(\mathbf{p}) \cdot \mathbf{M}_{\text{tan}} \cdot \mathbf{f}_{\text{map}}(\mathbf{p}), \tag{2}$$

where $\mathbf{f}_{\text{map}} : \mathbb{R}^3 \mapsto \mathbb{R}^4$ is another mapping function which is required to realize the different attachment types in Sections 3.3-3.5. Figure 4 illustrates the transformation of the detail point $\mathbf{p}$ to world space.

In summary, we require $\mathbf{f}_{\text{mt}}$, the user-defined functions $\mathbf{f}_{\text{uv}}$ and $\mathbf{f}_{\text{map}}$ as well as optionally the transformation matrices $\mathbf{M}_{\text{tex}}$ and $\mathbf{M}_{\text{tan}}$ to attach detail geometry. In the following subsections we will give the missing definitions of these functions.

3.2 The Transformation Field of a Surface

A surface in 3D space can be parametrized using two variables u, v, i.e., $(x, y, z) = \mathbf{f}(u, v)$. The tangent space plane can be computed from this surface parametrization for each (u, v) pair using partial derivatives in u and v as $\mathbf{t}_u = \frac{\partial \mathbf{f}}{\partial u}$ and $\mathbf{t}_v = \frac{\partial \mathbf{f}}{\partial v}$. These vectors define the directions of the principal axes of u and v in 3D space. Given the tangent vectors, we can compute the normal of the surface at position (u, v) as $\mathbf{n} = \mathbf{t}_u \times \mathbf{t}_v$ if the surface is not degenerated and if the parametrization on the surface is not pathological (i.e., $\mathbf{t}_u$ and $\mathbf{t}_v$ are not parallel). $\mathbf{n}$, $\mathbf{t}_u$, and $\mathbf{t}_v$ are then linearly independent and form a subspace in $\mathbb{R}^3$, which is aligned with the surface's tangent space. The tangent vectors and the normal are orthogonalized and normalized after their computation to obtain an orthogonal coordinate system. This is required to prevent a distortion of the attached geometry.

We extend the determined subspace with the position (x, y, z) on the surface to form a 3×4 linear transformation matrix. The matrix texture function which maps the surface parameter coordinates (u, v) to the corresponding transformation matrix is then defined by:

$$\mathbf{f}_{\text{mt}}(u, v) = \begin{pmatrix} t_{u,x} & n_x & t_{v,x} & x \\ t_{u,y} & n_y & t_{v,y} & y \\ t_{u,z} & n_z & t_{v,z} & z \end{pmatrix}. \tag{3}$$

3.3 Point Attachments

To attach a detail geometry to a point on a target surface (see Figure 2(a)), we first must define its goal position (u_g, v_g) in texture space. Then the corresponding transformation matrix is determined by Equation (3). By applying this transformation the source geometry is transformed on the surface and is oriented according to the principal axes of the subspace.

Since the goal position (u_g, v_g) of the detail geometry on the surface is predefined, the first mapping function is determined by

$$\mathbf{f}_{\text{uv}}(\mathbf{p}) = (u_g, v_g, 1)^T. \tag{4}$$

Hence, each point of the detail geometry has the same subspace (see Figure 2(a)). The detail geometry must be defined in the coordinate system of the tangent space. Since this geometry should only be attached at a single point, the second mapping function is determined by

$$\mathbf{f}_{\text{map}}(\mathbf{p}) = \begin{pmatrix} p_x & p_y & p_z & 1 \end{pmatrix}^T. \tag{5}$$

This means that the coordinates of the points of the source object are directly used to determine their positions in the subspace on the surface of the target object.

An example for point attachments is shown in Figure 1(a), where fur is fixed to a dress by this attachment type. The more complex area and curve attachments are described in the following subsections.

3.4 Area Attachments

In order to introduce an area attachment (see Figure 2(b)), we first need to define a mapping function $\mathbf{f}_{\text{uv}} : \mathbb{R}^3 \mapsto \mathbb{R}^3$ which yields valid texture coordinates $(u, v, 1)$ for each point of the detail geometry. Since the detail geometry is given in the coordinate system of the tangent space, the x, z coordinates correspond to the two tangent directions. Therefore, we the define the mapping function by a projection on the xz-plane:

$$\mathbf{f}_{\text{uv}}(\mathbf{p}) = (p_x, p_z, 1)^T. \tag{6}$$

If the resulting coordinates do not range from 0 to 1, we just scale the whole geometry accordingly to get valid texture coordinates.

In contrast to a point attachment we get a different tangent space matrix for each detail point. This means that each detail point has an own subspace which lies directly on the target surface (see Figure 2(b)). The distance of a point to this surface is defined by its y coordinate. Therefore, we get the final position for a point of the detail geometry by moving its corresponding surface position in normal direction by p_y. Mapping the point $\mathbf{p}$ by the function

$$\mathbf{f}_{\text{map}}(\mathbf{p}) = \begin{pmatrix} 0 & p_y & 0 & 1 \end{pmatrix}^T \tag{7}$$

yields the desired result. Recall that the first and the third column of $\mathbf{M}_{\text{mt}}$ correspond to the two tangent vectors, while the second and the fourth column correspond to the normal vector and the surface position (see Equation (3)).

3.5 Curve Attachments

The curve attachment is the most complex type. The source geometry must be attached to a curve on the deformed target surface (see Figure 2(c)). To define this curve, the user has to provide a parametric curve function $\mathbf{f}_{\text{c}} : \mathbb{R} \mapsto [0, 1]^2$. This function delivers two-dimensional coordinates, which can be directly used to determine the mapping function for the texture coordinates in Equation (1):

$$\mathbf{f}_{\text{uv}}(\mathbf{p}) = (\mathbf{f}_{\text{c}}^x(p_z), \mathbf{f}_{\text{c}}^y(p_z), 1)^T. \tag{8}$$

This means that we obtain a subspace on the target surface for each point on the curve (see Figure 2(c)). We use the z coordinate of the source point as parameter for the curve function. Hence, the curve is oriented along the vector t_v which is no limitation. For our examples (see Figure 1) we defined the curve function $\mathbf{f}_{\text{c}}$ using a cubic Bézier spline which is transformed to the desired position in texture space by $\mathbf{M}_{\text{tex}}$.

To define the offsets of a point $\mathbf{p}$ in the directions $\mathbf{t}_u$ and $\mathbf{n}$, we use the following mapping function in Equation (2):

$$\mathbf{f}_{\text{map}}(\mathbf{p}) = \begin{pmatrix} p_x & p_y & 0 & 1 \end{pmatrix}^T. \tag{9}$$

Since we used the z component of the source point as parameter for the curve, the corresponding value of $\mathbf{f}_{\text{map}}(\mathbf{p})$ is set to 0.

The mapping function of Equation (9) attaches a detail geometry to a curve on the target surface but does not align the geometry to the curve. If the source geometry should also be aligned to the curve, we use a different mapping function:

$$\mathbf{f}'_{\text{map}}(\mathbf{p}) = \begin{pmatrix} n_{\text{c}}^x \cdot p_x & p_y & n_{\text{c}}^y \cdot p_x & 1 \end{pmatrix}^T, \tag{10}$$

where $\mathbf{n}_{\text{c}} \in \mathbb{R}^2$ is the normal vector of the curve at p_z. This function determines the position of the point $\mathbf{p}$ in the plane which is orthogonal to the curve, where p_y is the height over the surface and p_x is the offset perpendicular to the curve.

3.6 Vertex Post-Processing

An important advantage of deferred warping is its seamless integration in the vertex transformation process. This provides the possibility to apply pre- and post-processing steps to the vertex data and yields a large flexibility. As an example, we demonstrate this flexibility in our fur rendering approach, which uses a post-processing step in order to emulate gravitational forces acting on the hairs (see Figure 1(a)). The hair geometry is transformed on the surface of the dress by using point attachments of small fur patches which are randomly placed and rotated. Without a post-processing step the hairs would all stand perfectly orthogonal to the surface which would look unrealistic. To obtain more realistic results, we therefore manipulate the vertex positions of a hair, which are given by Equation (2). First, the y coordinate of each point is reduced depending on its squared distance to the surface:

$$\mathbf{p}'_{\text{w}} := \mathbf{p}_{\text{w}} - \begin{pmatrix} 0 & c \cdot p_y^2 & 0 \end{pmatrix}^T, \tag{11}$$

where c is a user-defined value which scales the effect. After this step all hairs hang down but their lengths vary widely. We mitigate this problem by adapting the position $\mathbf{p}'_{\text{w}}$ so that it has the same distance to the corresponding surface point $\mathbf{p}_{\text{s}}$ as the original point $\mathbf{p}_{\text{w}}$:

$$\mathbf{p}''_{\text{w}} = \mathbf{p}_{\text{s}} + |\mathbf{p}_{\text{w}} - \mathbf{p}_{\text{s}}| \frac{\mathbf{p}'_{\text{w}} - \mathbf{p}_{\text{s}}}{|\mathbf{p}'_{\text{w}} - \mathbf{p}_{\text{s}}|}. \tag{12}$$

In this way a simple gravitational fur bending effect was introduced into the transformation. The result can be seen in the right image of Figure 9.

4 Implementation

Deferred warping is performed in a two pass rendering process at runtime. In the first rendering pass the matrix texture is updated. For an animated target geometry we update the texture per frame to keep track of the animation. To generate this texture we need to determine the transformation (see Equation (3)) of each vertex of the surface. This transformation consists of the vertex position as well as the normal and tangent vectors of the surface at the vertex. The position is already known and the required vectors can be easily obtained since state-of-the-art rendering engines often support dot product bump mapping. For this kind of bump mapping, tangent space data is needed and typically stored per vertex in conjunction with the surface normal. The shader transforms these vectors into world space. Additionally, tangent and binormal are orthogonalized and normalized, before a matrix is rendered into the texture. The matrix texture is then used in the second rendering pass in combination with the attachment mechanisms introduced in Section 3.

Each attachment type can be interpreted as an additional transformation step inside a vertex transformation stage of a renderer which is located between the local object transformation and the view and projection transformation. The transformation of the normal and tangent space of the source geometry is performed analogously to the vertex transformation. The three attachment types were implemented as vertex shader programs. Since only few values are required to control the mapping functions of the different attachments, we recommend to use bulk processing methods such as instancing on the GPU to maximize the geometry throughput. For this purpose we submit an array of parameters to the shader, which holds placement data for up to 256 geometry instances. Thus, larger blocks of object instantiations need to be broken into multiples of 256, giving a good speedup and still allowing arbitrary numbers of attached objects. A schematic view of the complete vertex transformation stage of our implementation is shown in Figure 5.

In our approach a problem arises when detail geometry tries to sample the matrix texture at positions not used by the target surface. Our solution to this problem is to initialize the matrix texture with a zero matrix. For a single vertex we can now detect, whether it is outside by checking the length of the orientation vectors of the matrix. If the length is smaller than one it is most likely for the vertex to have left the surface area. In our implementation we set a flag in this case within the vertex shader, the fragment shader can then discard all fragments, which are influenced by this vertex. This way, we can effectively detect and discard illegal states without much performance loss.

5 Results

We present our results here and in the accompanying video. All renderings were performed using an image-based lighting approach with ambient occlusion. All cloth simulations use the finite element method (FEM) of Etzmuss et al. [22]. For the simulation in Figure 7 we combined this method with an adaptive remeshing [23]. In our examples the

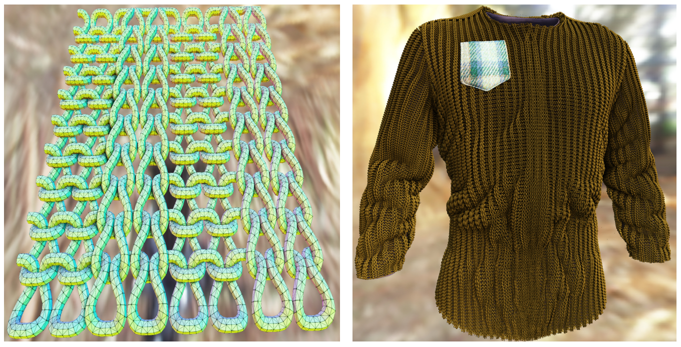


Fig. 6. Knitwear model (right) created by replacing the original cloth mesh by the knitwear pattern (left) using area attachment.

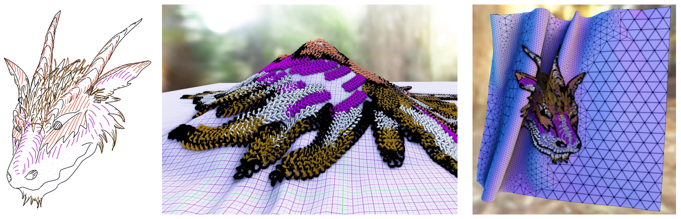


Fig. 7. Our implementation handles line drawings (left). Each curve is evaluated in the vertex shader to attach a source geometry repeatedly on the surface (middle). Deferred warping is independent of the mesh topology of the target surface and can therefore be directly used in adaptive cloth simulations (right).

matrix texture is assembled by four RGB textures using floating-point components with a resolution of 512×512 pixels.

Figure 1 shows results for point (a,c), curve (b,c), and area attachments (d,e) as well as combinations of them (c). The area attachment type even allows for replacing the surface completely. This can be used to replace a cloth model which consists of a simple triangle mesh by a complex knitwear model (see Figure 6) at interactive frame rates. The curve attachment shader can handle detail geometry that is repeated along the z-axis. The geometry is repeated n times via instancing and is mapped to the length of the curve (see Figure 7). The curves we use in the examples are cubic Bézier splines, which are read from scalable vector graphics (SVG) files. In Figure 7 (right) we also demonstrate that our approach can handle complex topology changes as they occur, e.g., in adaptive cloth simulations. If we implement $\mathbf{f}_{\mathrm{uv}}$ as a time-dependent mapping function to determine the texture coordinates, we can even animate the source geometry on a waving flag (see Figure 8).

Our novel method supports the usage of several attachment types at the same time and even allows to deform the attached geometry in a post-processing step (see Figure 9). This yields a large flexibility and allows the user to generate and to deform complex models at interactive frame rates. Therefore, our approach is well-suited for the usage in interactive virtual garment prototyping systems. Moreover, it can be easily integrated in such a system as geometry post-processing step of the garment simulation.

We tested the performance of our approach on an Intel Xenon X6550 (2,67 GHz) system with a NVIDIA GeForce GTX 580. For the test we attached a button geometry con-

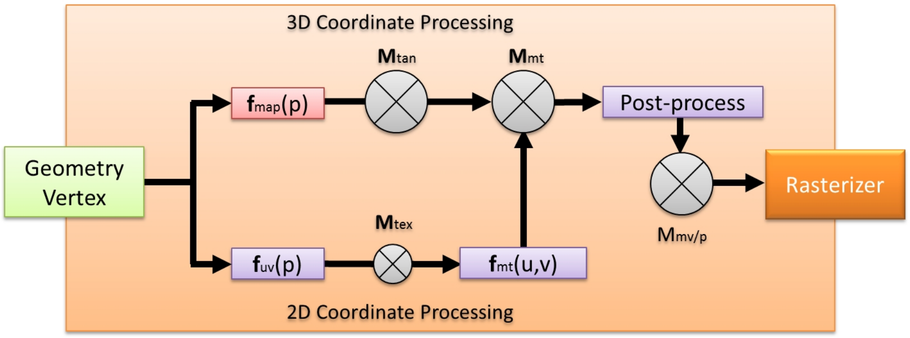


Fig. 5. Vertex transformation block diagram. The upper pipeline processes 3D coordinates from tangent space to world space. The lower pipeline processes 2D coordinates defining how the object is placed in texture space on the surface and therefore determines M_{mt}. The matrices $\mathrm{M}_{\mathrm{tex}}$ and $\mathrm{M}_{\mathrm{tan}}$ allow the independent positioning inside tangent and texture space. After an optional post-process of the transformed vertex it is passed to the standard model view transformation.

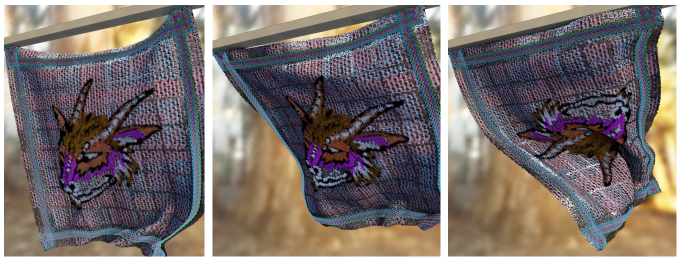


Fig. 8. Waving flag with a rotating logo as a demonstration of a time-dependent mapping function.

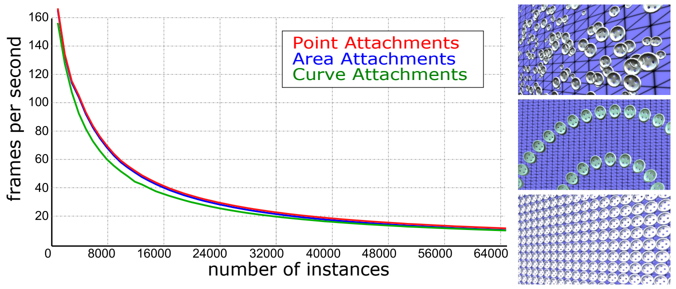


Fig. 10. Performance of deferred warping using an NVIDIA GeForce GTX 580 and massive instancing of a 928 triangles model in conjunction with our attachment types. The benchmark setups for point, curve and area attachments are shown on the right side (top to bottom). The benchmarks were performed by sequentially increasing the number of button objects on the surface. At the maximum, the pipeline handled 60.1 MTriangles. 65 measurements were taken per curve. GPU instancing was realized via the corresponding OpenGL extension. Point and area attachments have a similar performance. The curve attachment is computationally more expensive since it additionally has to evaluate parts of a curve in order to transform the vertices.

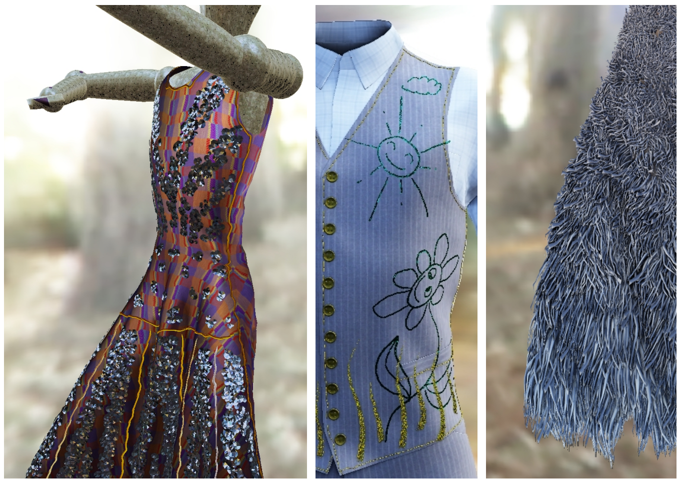


Fig. 9. Various attachment types on garment surfaces. Our approach allows to mix all types and to use them at the same time. Additionally, the technique can be easily extended to support additional features, e.g., to consider gravity in the fur example (right).

Image	Types	Triangles	Time (ms)
Fig1. a	points	25M	~ 37
Fig1. b	curves	10M	~ 20
Fig1. c	curves and area	6M	~ 13
Fig1. d	area	48M	~ 56
Fig1. e	area	650k	< 6

TABLE 1
Numbers of triangles of the detail geometries and computation times per frame of our algorithm for all examples shown in Figure 1.

taining 928 triangles up to 65000 times to a target surface resulting in up to 60.1 MTriangles. This test was performed with a point, an area, and a curve attachment shader. The results are shown in Figure 10. The performance values of the single attachment types are very similar and mainly depend on the number of attached elements. The curve shader has to evaluate cubic Bézier splines and is therefore a bit slower. Table 1 shows the number of triangles and the computation times for the scenarios in Figure 1. Please note: for all tests no culling or other acceleration techniques were used. The tests demonstrate how much performance can be expected using a naive implementation of the algorithm.

6 Discussion

Performance Considerations

Using a geometry based approach such as deferred warping has the advantage that it is possible to achieve a very high amount of detail albeit at the cost of performance. As demonstrated in Figure 10 (and as expected), there is a strong relationship between the speed and the amount of geometry rendered. In contrast, image-based approaches (e.g., Policarpo and Oliveira [6]) have the advantage that the required number of operations is limited by the number of visible pixels. However, ray casting approaches need to be directly supported by the used rendering system, whereas a pure geometry based approach can be integrated in practically all rendering systems. The resulting geometry can be post-processed or even exported without rendering output. A geometry based approach shifts the necessary computational power from pixels rendered to the complexity of the overall geometry processing. Thus, it is useful to reduce the amount of geometry, where it is not needed. Culling by leaving out invisible parts can improve the performance of the overall system by a great deal. A survey on occlusion culling techniques is given by Pantazopoulos and Tzafestas [24]. Due to the flexibility of our approach, culling techniques can be easily implemented even on single attachment element level. Therefore, using culling techniques in conjunction with our method is orthogonal. For this reason we do not discuss it in more detail in this work. Our performance tests were performed brute force to get reliable measurements. We did not use culling at all in our tests.

Flexibility

Looking at the flexibility of our approach, we have the advantage to work directly with the detail geometry data. This data is available in every stage of the process and can be manipulated in various ways, enabling effects such as bending fur or alternative attachment types. This breaks the limitation to only cover mapping of the detail geometry in all mentioned existing techniques. In terms of flexibility, deferred warping is therefore superior to these techniques. Additionally, it can be combined with every rendering technique and allows arbitrary pre- and post-processing of the detail geometry vertex data.

To prevent the self-intersections and inversions of the attached detail geometry, bends in the surface have to be limited to respect the height of the attached detail geometry. Additionally, deferred warping allows to reduce the amount of self intersections by low pass filtering the orientation field inside the matrix texture. This avoids sudden changes in the orientation of the applied objects. Since the matrix texture is basically an image, this filtering can be easily applied. However, to guarantee a penetration-free state more complex filter functions are required which is a topic for future work. Correctly handling self-penetration requires that the attached geometry is already considered in the generation process of the base mesh. Therefore, self-intersections cannot be completely resolved within our post-processing step which does not alter the base surface. Note that the problem of self-intersections is a common issue of all other surface detailing methods presented in the related work section.

7 Conclusion and Future Work

In this paper we introduced deferred warping, a novel approach for real-time deformation of 3D objects attached to an animated or manipulated surface. The method consists of two steps, creating the matrix texture containing the transformations and applying those to the detail geometry. One of its key advantages compared to previous work is the fact that detailed geometry is directly transformed, allowing additional manipulation in a post-processing step as well as efficient rendering using a standard rasterization-based pipeline. We demonstrated an implementation in the vertex shading stage of modern GPUs. Another key advantage is the flexibility to use different types of attachment which are typically not supported by existing techniques. This is exactly the flexibility which is required in virtual garment prototyping, where different appliqué, such as buttons or sequins, must be attached in different ways to garment models.

Our approach opens several directions for future work. First, we plan to combine deferred warping with procedurally generated surface geometry. This allows, e.g., to integrate more complex knitwear patterns in the virtual garment prototyping system. Applying deferred warping after the generation process could be a way to integrate procedurally generated surface geometry in real-time animations. Another topic we want to address in future is the missing collision feedback when using deferred warping in a simulation framework. Using the attached geometry directly for collision detection is too expensive for interactive simulations. To solve this problem one could introduce a collision geometry which approximates the final surface and transform this geometry simply by deferred warping.

Finally, we believe that our technique could greatly increase the visual complexity in various application domains including garments visualized in virtual try-on scenarios and computer games.

Acknowledgments

The work of Jan Bender was supported by the Excellence Initiative of the German Federal and State Governments and the Graduate School CE at TU Darmstadt. The research leading to these results has received funding from the European Commission's Seventh Framework Programme (FP7/2007-2013) under grant agreement n^{o} 285026.

References

[1] M. Fontana, C. Rizzi, and U. Cugini, "3D virtual apparel design for industrial applications," *Comput. Aided Des.*, vol. 37, no. 6, pp. 609–622, May 2005.
[2] N. Magnenat-Thalmann and P. Volino, "From early draping to haute couture models: 20 years of research," *The Visual Computer*, vol. 21, no. 8-10, pp. 506–519, 2005.
[3] R. Schmidt, C. Grimm, and B. Wyvill, "Interactive decal compositing with discrete exponential maps," *ACM Trans. Graph.*, vol. 25, no. 3, pp. 605–613, Jul. 2006.
[4] K. Zhou, X. Huang, X. Wang, Y. Tong, M. Desbrun, B. Guo, and H.-Y. Shum, "Mesh quilting for geometric texture synthesis," *ACM Trans. Graph.*, vol. 25, no. 3, pp. 690–697, 2006.
[5] C. Ma, L.-Y. Wei, and X. Tong, "Discrete element textures," *ACM Trans. Graph.*, vol. 30, no. 4, pp. 62:1–62:10, Jul. 2011.
[6] F. Policarpo and M. M. Oliveira, "Relief mapping of non-height-field surface details," in *Proc. I3D*, 2006, pp. 55–62.
[7] F. Neyret, "Modeling, animating, and rendering complex scenes using volumetric textures," *IEEE TVCG*, vol. 4, no. 1, pp. 55–70, 1998.
[8] R. d. Toledo, B. Wang, and B. Levy, "Geometry textures," in *Proc. SIBGRAPI*, 2007, pp. 79–86.
[9] X. Wang, X. Tong, S. Lin, S. Hu, B. Guo, and H.-Y. Shum, "Generalized displacement maps," in *Proc. EGSR*. Eurographics Association, 2004, pp. 227–233.
[10] S. D. Porumbescu, B. Budge, L. Feng, and K. I. Joy, "Shell maps," *ACM Trans. Graph.*, vol. 24, no. 3, pp. 626–633, Jul. 2005.

[11] S. Jeschke, S. Mantler, and M. Wimmer, “Interactive smooth and curved shell mapping,” in *Proc. EGSR*, 2007, pp. 351–360.
[12] R. Shen, “Animated volume texture mapping for complex scene generation and editing,” in *Proc. Smart Graphics*. Springer-Verlag, 2010, pp. 33–43.
[13] N. Ritsche, “Real-time shell space rendering of volumetric geometry,” in *Proc. Computer Graphics and Interactive Techniques in Australasia and Southeast Asia*. ACM, 2006, pp. 265–274.
[14] A. Brodersen, K. Museth, S. Porumbescu, and B. Budge, “Geometric texturing using level sets,” *IEEE TVCG*, vol. 14, no. 2, pp. 277–288, 2008.
[15] C. Yuksel, J. M. Kaldor, D. L. James, and S. Marschner, “Stitch meshes for modeling knitted clothing with yarn-level detail,” *ACM Trans. Graph.*, vol. 31, no. 4, pp. 37:1–37:12, 2012.
[16] S. Schein, E. Karpen, and G. Elber, “Real-time geometric deformation displacement maps using programmable hardware,” *The Visual Computer*, vol. 21, no. 8-10, pp. 791–800, 2005.
[17] G. Elber, “Geometric deformation-displacement maps,” in *Proc. Pacific Graphics*. IEEE Computer Society, 2002, pp. 156–165.
[18] T. Langer and H.-P. Seidel, “Higher order barycentric coordinates,” *Proc. EUROGRAPHICS*, vol. 27(2), no. 2, pp. 459–466, 2008.
[19] Y. Lipman, D. Levin, and D. Cohen-Or, “Green coordinates,” *ACM Trans. Graph.*, vol. 27, no. 3, pp. 78:1–78:10, 2008.
[20] T. Akenine-Möller, E. Haines, and N. Hoffman, *Real-Time Rendering 3rd Edition*. A. K. Peters, 2008.
[21] K. Hormann, B. Lévy, and A. Sheffer, “Mesh parameterization: theory and practice,” in *ACM SIGGRAPH courses*. ACM, 2007.
[22] O. Etzmuss, M. Keckeisen, and W. Strasser, “A fast finite element solution for cloth modelling,” in *Proc. Pacific Graphics*, 2003, pp. 244 – 251.
[23] J. Bender and C. Deul, “Adaptive cloth simulation using corotational finite elements,” *Computers & Graphics*, vol. 37, no. 7, pp. 820 – 829, 2013.
[24] I. Pantazopoulos and S. Tzafestas, “Occlusion culling algorithms: A comprehensive survey,” *J. Intell. Robotics Syst.*, vol. 35, no. 2, pp. 123–156, Nov. 2002.

Martin Knuth is a researcher at Fraunhofer Institute for Computer Graphics, Darmstadt, Germany. He received his diploma in computer science from the Technische Universität Darmstadt. His research interests include interactive simulation and visualization suitable for virtual prototyping of garments. Contact him at martin.knuth@igd.fraunhofer.de.

Jan Bender is a assistant professor of computer science at the Graduate School CE, TU Darmstadt. He received his diploma, PhD and habilitation in computer science from the University of Karlsruhe. His research interests include interactive simulation of multibody systems and deformable solids, collision handling, fracture, fluid simulation and real-time visualization. Contact him at bender@gsc.tu-darmstadt.de.

Michael Goesele is a professor of computer science at TU Darmstadt. He holds a diploma in computer science from Ulm University and a doctorate from Saarland University and the MPI Informatik. His research interests include computer graphics, computer vision, and massively parallel computing. Contact him at michael.goesele@gris.informatik.tu-darmstadt.de.

Arjan Kuijper is a professor of computer science at TU Darmstadt and a staff member at the Fraunhofer Institute for Computer Graphics Research (IGD). His research interests include mathematics-based methods for image processing, pattern recognition, computer vision and graphics. Kuijper received a habilitation from TU Graz’s Institute for Computer Graphics and Vision and a doctorate from Utrecht University. Contact him at arjan.kuijper@igd.fraunhofer.de.