Eurographics Symposium on Rendering 2009
Hendrik Lensch and Peter-Pike Sloan
(Guest Editors)

Volume 28 (2009), Number 4

Texture Design and Draping in 2D Images

H. Winnemöller[1] A. Orzan[1,2] L. Boissieux[2] J. Thollot[2]

[1] Adobe Systems, Inc. [2] INRIA – LJK

Abstract

We present a complete system for designing and manipulating regular or near-regular textures in 2D images. We place emphasis on supporting creative workflows that produce artwork from scratch. As such, our system provides tools to create, arrange, and manipulate textures in images with intuitive controls, and without requiring 3D modeling. Additionally, we ensure continued, non-destructive editability by expressing textures via a fully parametric descriptor. We demonstrate the suitability of our approach with numerous example images, created by an artist using our system, and we compare our proposed workflow with alternative 2D and 3D methods.

Categories and Subject Descriptors (according to ACM CCS): Picture/Image Generation [I.3.3]: —Graphics Utilities [I.3.4]: Graphics Editors—

1. Introduction

Textures play a vital role in human perception and have found widespread use in design and image synthesis from 3D scenes. Gibson [Gib50] first described the importance of textures for the perception of optical flow, and many authors have since investigated how texture variations are linked to the perception of material properties and surface attributes, most prominently shape [CJP93]. Artists and designers commonly employ textures to embellish objects, such as furniture, clothes, or walls, with designs and color (Figure 1).

Despite this prevalent use of textures, there exist only few methods for arranging and manipulating them in 2D images, a process we call *texture draping*. Purely 2D methods are often intuitive, but require a lot of manual editing, as they lack high-level tools to suggest an underlying shape to be textured. Conversely, purely 3D methods provide such high-level tools, and allow for very exact shape simulation, but require 3D models, which are difficult and time-consuming to construct. To complicate matters, most texture distortions fall into two broad categories: *Shape-based* distortions, which are due to the geometry on which a texture rests (Figure 1), and *design-based* distortions, which are due to irregularities in the texture-design itself (Figure 7). Obviously, different tools are necessary to address both types of distortions.

To address these issues, we propose a 2.5D solution, which marries the advantages of previous approaches, while minimizing their drawbacks. Specifically, we support sketch-based *minimal* shape-modeling. That is, artists design normal fields of the minimum complexity necessary to achieve the desired image-space effect. In addition, artists can locally manipulate 2D texture-coordinates using a rubber-sheet technique. Compared to 2D warping techniques, this allows for better control over affected areas, without requiring careful masking. Together, these two approaches allow for texture draping of a wide variety of implied shapes and texture designs, as demonstrated by the examples throughout this paper.

Figure 1 shows a typical application example. A designer wants to quickly prototype different designs for a dress. She sketches out the outline of the dress, as well as major folds and creases (Figure 1a), a process she is used to from her traditional design workflow. For each curve she has drawn, she can define normals and texture-coordinate (u,v) parameters. Once she has completed her texture draping, she can substitute any number of texture swatches to test her designs (Figure 1b).

We make several contributions within our 2D texture design and draping system. Compared to previous works, we focus on art-creation workflows from scratch, i.e. where no existing shape model or image exists. We identify two types of complementary texture distortions that should be supported in such a system, and provide tools to intuitively affect these distortions via normal and (u,v) coordinate manipulations. We demonstrate how these distortion parame-

(a) (b)

Figure 1: Design Example. *(a) Our draping approach supports editing operations that allow for precise placement of textures in an image. (b) Given a set of texture-edit curves, we can apply any number of textures (here, composited over a photograph). The inset texture swatches are also designed using our system, except for the rightmost swatch, which is a bitmap texture.*

ters can be encoded and manipulated within the framework of a fully parametric image description, thus gaining many benefits of a vectorial representation. By using the same representation for texture draping, as well as the texture description, we can reuse the normal information to achieve simple shading effects, including macro-structure of texture elements. Finally, we demonstrate in a limited case-study that our system performs favorably when compared to alternative 2D and 3D approaches.

2. Related Work

Applying textures in 2D Numerous authors have addressed the problem of applying textures in 2D images with a variety of approaches.

Many of these works focused on synthesizing natural, realistic-looking textures [RLA*06, FH07, KEBK05, LH06]. As such, they allowed users to specify the location in an image where a texture was to be synthesized, but they did not focus on providing tools to precisely adjust the texture distribution within textured regions. In fact, the purpose of many of these works was to free the user from the burden of explicit texture control. Nonetheless, explicit control is necessary if textures are to be draped in such a way as to suggest shape.

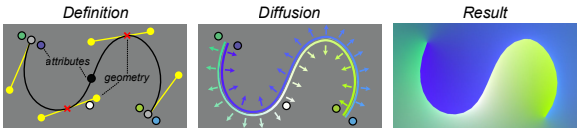
The automatic system of Fang and Hart [FH04] provided such explicit shape control via a normal map that was extracted from a source image using shape-from-shading techniques. This approach is a good choice if a smooth-shaded, non-textured image is already available. We assume, though, that in many design situations this is not the case.

Liu et al. [LLH04] proposed a system that allowed users to texture images, even if these images already contained existing regular textures. For this, they required users to manually mark all texels (texture elements) in the source image, which they used to deform a regular grid mesh. This mesh could then be re-textured and re-lit, using a shading map extracted from the original image. As above, we cannot assume the availability of a suitable source image. Additionally, our goal is to provide an artist with high-level tools that do not require the manual placement of individual texels.

Image warping, such as the techniques described in [Wol94], can also be used to locally distort textures to suggest shape, but such an approach requires a lot of manual masking to protect areas from being distorted inadvertently. In Section 6 we compare our technique with the image warping method implemented in PHOTOSHOP's *Liquefy* tool.

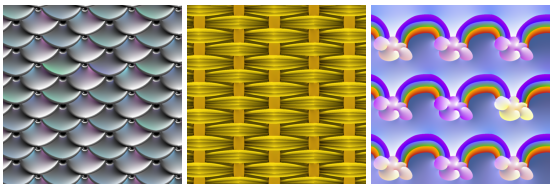
Applying textures in 3D The technical aspects of mapping 2D textures onto 3D surfaces and their subsequent projection onto the image-plane are well understood [Hec86, SKvW*92, LÓ1].

Even so, manual placement of textures onto 3D surfaces for anything but simple geometric objects, like planes or spheres, is still difficult and subject of recent research [GDHZ06]. More importantly, the need to create 3D models to be textured is the main drawback of this approach, as 3D modeling is notoriously difficult [IH03, NSACO05, BBS08]. Furthermore, many design situations are conceptualized entirely in 2D, so creating a 3D model that exactly matches the intended 2D design requires a difficult reverse modeling process (Viz. Section 6).



(a) (b) (c)

Figure 2: Diffusion Curves. *A diffusion curve image is generated by: (a) defining geometric curves, and attributes along those curves; (b) interpolating attributes along the curves, then diffusing them outwards; (c) to obtain the final image.*



(a) (b) (c)

Figure 3: Texture-map Examples. *Examples of textures designed with our system: from realistic (a) to artistic (c).*

3. System Design Considerations

Given the many available texture editing operations, texture generation approaches, and image descriptors, we discuss here the motivating factors that prompted our specific system design to best facilitate texture design and draping in 2D images.

Draping Features Since there exist simple modeling tools for trivial geometry (planes, boxes, spheres, etc.) we consider the texture draping problem for imagery comprising these shapes solved. Instead, we focus on imagery with organic shapes, and flowing features (both 2D and 3D), which are generally difficult to model, or even simulate.

By taking inspiration from garment and furniture design (Figure 1a) we identify the following prevalent types of draping features:

- *Shape contours* - delineate the extent of textured regions
- *Creases & folds* - exhibit sudden change of normals
- *Occlusions* - where one piece of material overlaps another
- *Seams* - where one piece of material abuts another

Note how all of these features describe *discontinuities* of one type or other, i.e. features are assumed to be mostly smooth except for a few distinct lines and curves.

Texture Type Texture generation has been the focus of much research, and common generative approaches include *synthesis* [SCSI08], *procedural* [EMP*03], and *manual* [IMIM08] . While all of these methods can be used to generate texture maps for our system, we concentrate our attention on manual methods, specifically *regular*, or *near-regular* textures [LLH04]. The reason for this is that, due to their regular structure, texture deformations are more easily visible in these types of textures [CJP93], thus better illustrating our approach.

Parametric Descriptor To allow an artist or designer the greatest level of convenience, control, and flexibility, we argue that a parametric (vector-based) texture draping approach is important, particularly as this permits intuitive and continued editing of the textured image.

Many existing vector formats already support textures, such as SVG, CorelDraw™, Flash™, or Adobe Illustrator™. The types of supported textures generally include bitmaps, procedural textures, and tiled vectorial patterns. However, the draping support in these formats is generally limited to planar translation, rotation, and scaling. While Illustrator™ recently introduced 3D-based texture mapping, this requires a full 3D model, and does not cater for low-level texture draping control.

Texel Descriptor Another format consideration is *visual complexity & quality*. While bitmap textures can achieve arbitrary complexity, they are limited to a specific resolution, which can produce undesirable stretching and blurring when distorted too much. Vectorial patterns, on the other hand, are resolution-independent, but lack visual complexity.

Recently, two image descriptors have been proposed which overcome these limitations, namely *Gradient Meshes* (Illustrator, [SLWS07]), and *Diffusion Curves* [OBW*08]. Both are parametric representations, and allow for arbitrarily complex image depiction. Gradient meshes define geometric control points in a regular mesh consisting of Ferguson patches. Colors defined at each node are interpolated along patch boundaries and the interior. As such, gradient meshes are highly compatible with the texture-mesh approach of Liu et al. [LLH04], but they also share the same requirement for intensive manual editing.

Diffusion curves (DCs) (Figure 2) are geometric curves with various appearance attributes attached, namely *color* for each of the sides of the space that a curve divides, and *blur* magnitude to define how quickly color varies across the curve. Even though this approach only sparsely defines image properties explicitly along a few user-specified curves, the appearance of all other regions is implicitly defined by interpolating attribute values linearly along each curve, and then diffusing the attributes throughout the entire image using a Poisson solver [WSTS08]. For our purposes, Diffusion Curves bear the following advantages:

- *Draping* - DCs are defined along feature discontinuities in an image, just like the draping features we intend to model.
- *Appearance* - DCs can depict arbitrarily complex image content, from realistic to abstract. As such, they are well suited to form visually appealing textures.

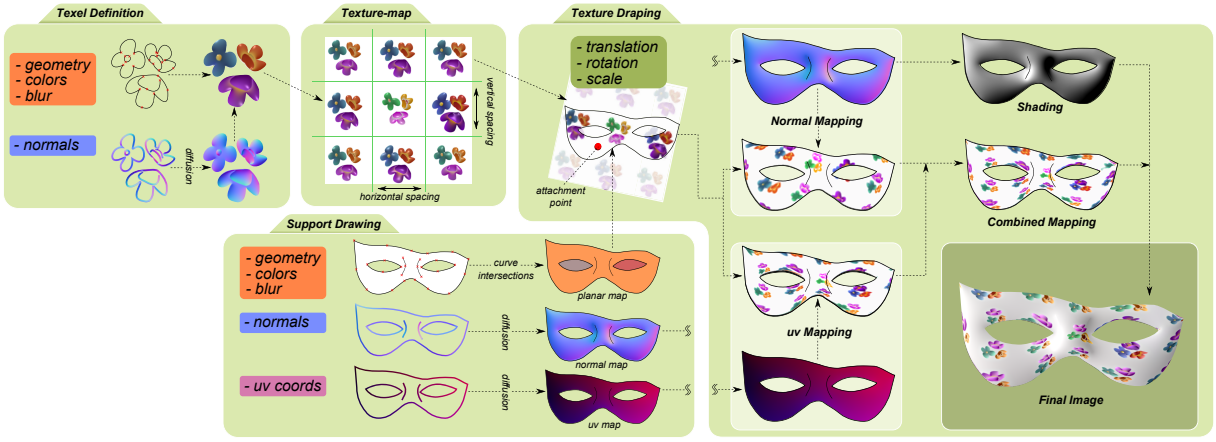


Figure 4: Overview. *Our texture design and draping system starts with the creation of **texels** and a **support drawing**. Texels are combined into **texture-maps**. The texture-map is **draped** over the image using either **normal** controls, **(u,v)** controls, or a combination of both. The final image can be optionally **shaded** automatically using normals, or manually via α-blending with the support-drawing's diffusion curve colors.*

- *Vector-format* - As a vectorial descriptor, DCs share the common advantages of vector formats, such as resolution-independence, compactness, editability, etc. [ICS05].

Note how the above list employs DCs both for texture draping, as well as the texel descriptor (appearance). This points to additional advantages in terms of a unified set of interaction paradigms, and implementing both design and draping aspects with related code components.

Considering the discussed design choices, Figure 4 illustrates our approach: DCs are first used to create both a support drawing and numerous texels. The texels are then arranged in texture-maps, and the texture-maps are attached to regions of the support drawing. Shape distortions (via normal controls) and design distortions (via (u,v) texture-coordinates) are combined to create the final textured image.

4. Texture Parameters

As noted above, there are two main types of texture distortions that are relevant for the workflows we facilitate: shape and design distortions. This section explains the main causes for these distortions and argues for parameters to control them.

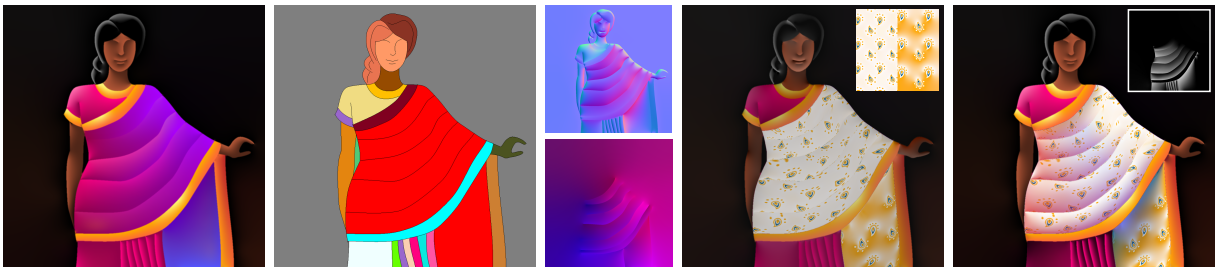
Shape Distortions Cumming et al. [CJP93] identified three perceptual parameters to deduce shape from texture variations: *compression*, referring to the projective change in texel shape when mapped onto a surface non-orthogonal to the viewer, *density*, and *perspective*. The strongest shape cue, compression, can be produced by distorting a texture according to the surface normal of an object. Note, that specifying surface normals is a simpler problem than specifying a full 3D shape. We thus allow users to specify normal-constraints at any points along a Diffusion Curve. As with color attributes, normals can be specified on either side of a curve. Another advantage of using normals is that they can be used for simple shading effects, as in Figure 8a. Furthermore, since the texture-maps themselves are comprised of extended diffusion curves (Section 5.1), we can easily model textures with fine macro-structure [LM99], such as wood bark, basket weave, or scales (Figs. 3a,b).

Design Distortions To easily specify distortions that are not strictly due to plastic shape, for example, the hair or the tail of the mermaid in Figure 7, we enable the user to locally offset (u,v) texture-coordinates. This approach is also useful to capture 2D shape distortions that are not modeled by surface normals, such as the undulating shape of a snake, or the holes in a mask (Figure 4). As above, we permit (u,v) distortions to be specified bilaterally at any point along a Diffusion Curve.

With these additions, each extended Diffusion Curve is defined as:

$\mathrm{P}[]$	array of $(x,y,tangent)$ tuples specifying the Bézier spline
$C_l[], C_r[]$	arrays of (r,g,b,α,t) for the left- and right-side colors and their parametric positions on the curve
$\Sigma[]$	array of (σ,t) blur values
$N_l[], N_r[]$	arrays of (x,y,z,t) normals on each side
$U_l[], U_r[]$	arrays of (u,v,t) texture coordinates

In addition to the above *distortion* parameters, we need to provide parameters that describe each textured *region*, including: The bounds of a region, the texture-map *id* to be applied to that region, and a local texture coordinate-frame. We also add a surface inflation/deflation toggle for user convenience (Section 5.1).



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

Figure 5: Complete Example. *(a) The support drawing with colors. (b) Implicitly defined planar map regions. (c) Normal map, top. (u,v) map, bottom. (d) Inset texture, applied to the support drawing. (e) Inset α-map, applied to manually shade the image.*

5. Implementation

The following implementation discussion is summarized in Figure 4, and closely follows a typical artistic workflow for our system. To be able to drape a texture, we need to first create a texture-map.

5.1. Creating a Texture-Map

For the regular or near-regular textures used in this paper, we let the user specify one or more texture elements (texels) which are replicated throughout a grid.

An individual texel is specified as a regular Diffusion Curve, described by Orzan et al. [OBW*08]. The user draws geometric curves in the plane, and assigns colors and blur values to achieve the desired look.

Given our normal extension, the user can additionally specify normal attributes along each curve, to create a bumpmap-like effect. For this, the user selects a point along a curve and chooses to add normal constraints on one or both sides. These normal constraints are linearly interpolated along each curve and diffused. Unlike other attributes, whose coordinates are generally mutually independent, we must re-normalize normals after diffusing them.

Since many organic shapes that we want to model are blob-like, we simplify normal specification for these shapes by providing an *auto-inflation* mode, similar to previous surface inflation methods [Joh02, JC08]. In this mode, the user only needs to specify the rotation of the curve's instantaneous normal around the curve's instantaneous tangent. This limits the specification from three degrees of freedom, to one. Furthermore, the user needs to specify whether the automatic mode inflates (out of page) or deflates (into page) the suggested shape.

By default, a texel is replicated throughout a square grid, thus creating a *regular texture*. The user can adjust the horizontal and vertical spacing of the unit cell to adjust the texel distribution.

To support more creative flexibility, we provide a pseudo-random perturbation mechanism, as follows. The user chooses any texture-map cell to modify it. The system then instantiates a copy of that cell's texel for the user to adjust. The user can modify any of the extended Diffusion Curve parameters or add/delete entire curves. In the latter case, the system also adds/deletes these curves to all other texels to retain a one-to-one correspondence between all texels defined for the texture-map. The system then randomly interpolates texel parameters throughout the entire grid, in the spirit of Baxter and Anjyo [BA06], to create a more varied looking texture-map. This interpolation is stable for all but extremely different source texels (i.e. large user distortions). If the user is dissatisfied with a particular random texel, she can instantiate and modify it manually. An example of a randomized texture-map is shown in Figure 3c.

5.2. Draping a Texture-Map

To drape a texture-map in an image, the user first needs to design a drawing on which to apply the texture, called the *support drawing*. This process is the same as that of drawing a standard Diffusion Curve image, or designing a single texel (Fig. 5a).

Orzan et al.'s original Diffusion Curves did not have a notion of *regions*. For texturing, however, it is sensible to define regions to which texture parameters can be attached. Our system creates regions implicitly by computing a planar map arrangement from the Diffusion Curve geometry with the help of the CGAL (`http://www.cgal.org`) library, as shown in Figure 5b.

The user can click on any planar map region and *attach* a texture-map to it. The coordinate frame for the texture-map is defined by the position of the attachment point. The user can translate the texture-map in the image plane by dragging the attachment point in the texture region. Planar scaling and rotation assume the attachment point as the local center of origin. When the drawing is modified and the planar map

(a) (b)

Figure 6: Parallax Mapping:. *Examples of texture draping using only parallax mapping. (a) is shaded using the normal map and (b) is shaded manually.*

(a) (b)

Figure 7: Direct Texture Control. *Here, an artist managed to skillfully drape the texture to suggest curly hair, flow the scales-texture along the mermaid's tail, and apply fins to the tail's tip - all by direct (u,v) manipulation. (a) Textures only. (b) Manual shading applied to textures.*

updated, the attachment point's image-space coordinate dictates which new region corresponds to the texture. If several texture attachment points fall within the same region, the user simply selects the active texture or drags other textures to different regions.

Normal Mapping As above, the user can specify *normals* along DCs, to suggest the underlying shape of the texture region (Fig. 5c, top). In addition to simple shading, however, the normal information is used to distort the attached texture. We use the parallax mapping technique [Wel04], which warps the texture to give the impression of parallax foreshortening. Given a texture-map applied to a flat polygon (in our case the image rectangle), parallax mapping offsets each texture coordinate to suggest complex surface shape (Figure 6). In practice, a height-field needs to be computed from the normal-map. As in other parts of our system, we solve a Poisson equation for this purpose.

For an image pixel (x,y) with the initial texture coordinate (u_0,v_0) and height value h, the normal-adjusted texture coordinate is $(u,v) = (u_0,v_0) + h \cdot V_{x,y}$, where $V_{x,y}$ are the x and y values of the eye vector.

For automatic shading, the normal map of each texel is simply added to the normal map of the support drawing, and re-normalized.

UV Mapping For design distortions, and shape distortions that cannot be modeled with a normal field (Figure 7), the user can locally offset (u,v) *coordinates* (Fig. 5c, bottom). Our implementation is inspired by rubber-sheet techniques, where the user can specify and pin exact texture coordinates at chosen control-points along a curve, and the remaining texture stretches in-between to fit the constraints. As elsewhere, this is achieved with linear constraint interpolation and Poisson diffusion.

For convenience, the user can use a sampling option to initialize the (u,v) coordinates. The system automatically computes default positions and values to create the least possible distortion in the texture, while adding as few control points as possible along the chosen curve. To do so, we use the Douglas-Peucker algorithm [DP73] to find a set of points that approximate the selected Bézier curve and place (u,v) coordinates on them, so that the texture lies flat in the image space.

Normal mapping and (u,v) mapping can easily be combined to emulate complex folds and ripples. In that case, the (u,v) coordinates are used as initial coordinates for the normal mapping. This effect was used in several figures in this paper. For example, the sari in Figure 5 uses (u,v) mapping to suggest draping of the cloth, and normal mapping to add folds to the fabric.

5.3. Rendering

Even though our prototype implementation does not offer user conveniences such as specialized widgets and proper UI design, it is intended to demonstrate the utility of our approach. As such, interactive system performance is paramount.

Similar to Orzan et al.'s [OBW*08] rendering approach, all texture-maps are rasterized and cached as OpenGL textures, to enable real-time visual feedback during texture design and draping. The final output of our system is still resolution-independent as the rasterization is recomputed for any given zoom level. To minimize stretching artifacts when textures are inflated towards the viewer, we rasterize texture-maps at the maximum height after normal integration.

Another advantage of caching textures as bitmaps is an effective decoupling of texture *draping* from texture *representation*. In fact, our draping approach can be applied to any bitmap texture, either existing, or generated procedurally or

(a) (b)

Figure 8: Shading Effects. *(a) A realistic shading effect based on the normals of both the texture-map and the support drawing. (b) An artistic shading, realized with manual shading controls.*

via synthesis. Fig. 1b (right) shows an example of a draped bitmap texture. As above, the resolution of such a bitmap should be high-enough to avoid stretching artifacts.

As noted above, almost all extended Diffusion Curve attributes are linearly interpolated and diffused via a Poisson solver. To ensure optimal interactivity of our system, we cache all attribute diffusion results for the current zoom level, and only recompute the attribute that the user currently modifies. Combining the diffused maps of different attributes (e.g. texture coordinates from (u,v) and normal distortion) is comparatively cheap and does not add much overhead.

The normal parameter of our extended DCs allows for simple shading computations, exemplified here with a Phong-based model. While this automatic shading proved convenient, especially to emphasize macro-structure of texels, it is rather limited. A more complex shading model is not only beyond the scope of this paper, but, in our opinion, the wrong approach. Orzan et al. [OBW*08] have already shown that Diffusion Curves are capable of depicting complex gradients and scene shading. In our system, we therefore allow the user to accept the automatic shading provided by the system, or paint over the automatic shading result with standard Diffusion Curve colors. To enable this choice, we simply add an α-channel to the color attribute, which effectively allows an artist to locally mix both manual and automatic shading, as she sees fit (Figs. 7b, 5e, 8b).

6. Evaluation

To evaluate the feasibility and utility of our system to create textured 2D images from scratch, we charged a professional digital artist, proficient with modern 2D and 3D art creation tools, with creating a number of different texture types and designs, and to report on her experience with our system. The figures in this paper represent some of this artist's work with our system.

Timing Table 1 lists detailed timings for several figures in this paper. Durations are listed separately for the creation of the support drawing, the texture draping, and the texel generation for that figure.

Figure	Support	Draping	Texels
Dress, Fig. 1	10 min(G)	30 min	5-30 min
Overview, Fig. 4	5 min (G)	20 min	40 min (G+C+N) for 2 texels
Sari, Fig. 5	1 h (G+C)	65 min	20 min (G+C)
Sari, Fig. 6b	1 h (G+C)	1h (N)	7 min (G+C)
Mermaid, Fig. 7	1 h (G+C)	45 min (UV)	5 min each (G+C)

G–*Geometry* ; C–*Color* ; N–*Normals* ; UV–*Tex. Coords.*

Table 1: *Timings for selected Figures*

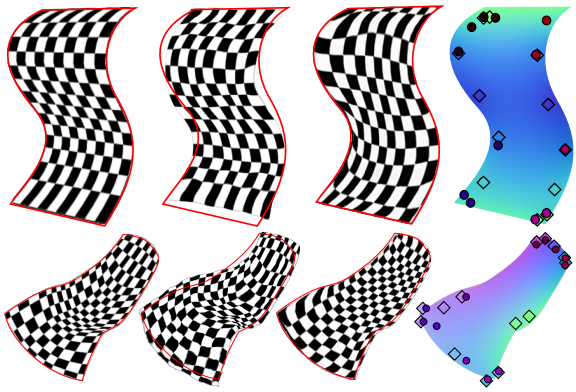
Feedback After using our system, we asked the artist to give us feedback about her experiences, both positive and negative. On the positive side, she noted that she found the *interactions* were very intuitive, as they were approaching a traditional design-on-paper workflow. On the negative side, she complained about various *interface* aspects. For example, adjusting (u,v) coordinates to realize her intentions was initially too time-consuming. After we implemented the automatic (u,v) initialization (Sec. 5.2), she found the system much easier to use.

Alternative 2D Draping To compare our system against 2D warping approaches, we asked the artist to design two simple 3D reference shapes in a 3D modeling package and texture them with a checkerboard texture (Fig. 9a). She then had to replicate the 3D rendering output as closely as possible with our system (Fig. 9c), and PHOTOSHOP's *Liquefy* tool (Fig. 9b). Note, that this required not merely giving a good impression of shape, but to match each texel – a much more difficult task.

The two 3D shapes she designed were a simple *S-shape*, and a more complex *Seat-shape*. Using our system for the *S-Shape*, she spent $1'29''$ on the support drawing, $2'27''$ on setting normals, $4'25''$ on adjusting automatically placed (u,v) points, and $1'25''$ on adding and setting additional (u,v) points, for a total of just under ten minutes. Fig. 9d shows the normal control points (*diamonds*) and (u,v) control points (*circles*) the artist specified. She commented that much of the time spent on (u,v) adjustments was due to difficulties with not visualizing the texture-map in our texture-coordinate editor. Timings for the Seat-*shape* were similar, but added up to only $9'31''$, indicating that labor is proportional to the 2D complexity of the suggested shape, not its 3D complexity.

Using the *Liquefy* tool, the artist started with a rectangular checkerboard texture and spent $10'57''$ deforming it on the *S-Shape*, and $32'22''$ on the *Seat-Shape*. As evident in Fig. 9b, the artist did not manage to control the exact contours of the shape. She commented that the warping approach was tedious due to the requirement of frequent and careful masking, and constantly changing the radius of the distortion tool.

3D Texturing We also asked the artist to compare our system with standard 3D texturing. As reference, the time to model and texture the above 3D *S-Shape* was about half that for replicating it using our system. The timings for the *Seat-*



(a) 3D (b) *Liquefy* (c) Ours (d) Controls

Figure 9: Comparison with *Liquefy* Tool. *Top row: S-Shape. Bottom row: Seat-Shape. (a) 3D result. (b) Liquefy Tool. (c) Our system. (d) Control Points & normal-map.*

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

Figure 10: Comparison with 3D Modeling. *(a) Draping template. (b) as (a) with checkered texture. (c) 3D model with checkered texture. (d) 3D model with texture from (a).*

shape were comparable. As these numbers favor our system for complex 3D shapes, and since we envision our system being used in design workflows that are conceptualized in 2D, we performed a second test with the artist, complementary to the one above. Here, we replaced the sari texture in Figure 10a with a checkerboard texture and asked the artist to create a 3D model to achieve the same image.

The artist took $3h45'$ to generate a 3D model of the draping. As Fig.10c shows, this included only the sari but no background elements. She worked for an additional hour to adjust (u,v) coordinates using a professional skinning tool. This is compared to two hours total for our system, including geometry and color for background elements. When asked about her experience, she said that she favored 3D modeling for simple geometric shapes, but preferred the natural 2D design approach of our system for the complex shapes of drapings and folds that she created. She also pointed out that she was unaware of a straightforward 3D method to create the artistic design of the mermaid's hair in Fig. 7.

7. Conclusion

As shown by the many example drawings in this paper, and an artist's feedback, our system is capable of designing appealing looking textures, and draping these textures on a 2D image with intuitive and flexible controls.

We acknowledge several limitations of our implementation. For very simple shapes, a 3D modeling system is quicker to use. In general, our system is not intended to replace accurate 3D systems, but rather to allow for quick and convenient prototyping of complex texture draping designs. Additionally, some aspect of our interface design proved to be cumbersome. While we hope to streamline the interface in the future, we feel this does not detract from the fundamental interactions, which an artist using our system quickly learned and mastered.

Currently, our system only supports regular or near-regular textures. In Section 5.3, we note that any texture-generation approach which outputs bitmaps can be used in our draping system. We want to investigate several such approaches, and determine what additional user-parameters are necessary to control different types of textures.

Given the resolution independence of a vector-based representation, we think that level-of-detail considerations and hierarchical textures, akin to Han et al. [HRRG08] are worth investigating.

Finally, as we focused on manual creation and draping of textures in this work, we did not address how draping parameters could be automatically extracted from source images. Such an approach would face the same challenges as previous works [FH04, LLH04], namely texture-analysis and shape-from-shading (normals). Both of these problems are still active research areas in computer vision. So, instead of a fully automatic solution, we believe that a user-assisted approach might be feasible [WSTS08].

In summary, we have presented a system with which a user can design vectorial textures, including complex color appearance and macro-structure. Employing the same interaction paradigm as for texture and support-drawing design, the user can drape textures over a 2D image, by manipulating normals of a suggested shape, and (u,v) parameters. Finally, the user can synthesize simple shading over the image automatically, or manually shade the image for dramatic effect. The proposed draping parameters and tools allow for the creation of realistic looking texture designs (Fig. 1) as well as purely artistic designs (Fig. 7).

Acknowledgments

Alexandrina Orzan is supported by a grant from the European Community under the Marie-Curie project MEST-CT-2004-008270. LJK is a joint research laboratory of CNRS, INRIA, INPG, Université de Grenoble I and Université de Grenoble II.

References

[BA06] BAXTER W., ANJYO K.: Latent Doodle Space. *Computer Graphics Forum 25*, 3 (2006), 477–485.

[BBS08] BAE S.-H., BALAKRISHNAN R., SINGH K.: ILoveSketch: As-natural-as-possible sketching system for creating 3D curve models. In *UIST '08* (2008), ACM, pp. 151–160.

[CJP93] CUMMING B., JOHNSTON E., PARKER A.: Effects of different texture cues on curved surfaces viewed stereoscopically. *Vision Research 33* (1993).

[DP73] DOUGLAS D., PEUCKER T.: Algorithms for the reduction of the number of points required for represent a digitzed line or its caricature. *Cartographica: The International Journal for Geographic Information and Geovisualization 10*, 2 (1973), 112–122.

[EMP*03] EBERT D. S., MUSGRAVE F. K., PEACHEY D., PERLIN K., WORLEY S.: *Texturing and Modeling: A Procedural Approach*, 3rd ed. Morgan Kaufmann, 2003.

[FH04] FANG H., HART J. C.: Textureshop: texture synthesis as a photograph editing tool. In *SIGGRAPH '04: ACM SIGGRAPH 2004 Papers* (2004), ACM, pp. 354–359.

[FH07] FANG H., HART J. C.: Detail preserving shape deformation in image editing. *ACM Trans. Graph. 26*, 3 (2007), 12.

[GDHZ06] GINGOLD Y. I., DAVIDSON P. L., HAN J. Y., ZORIN D.: A direct texture placement and editing interface. In *UIST '06* (2006), ACM, pp. 23–32.

[Gib50] GIBSON J.: *The perception of the Visual World*. Houghton Mifflin, 1950.

[Hec86] HECKBERT P. S.: Survey of texture mapping. *IEEE Comput. Graph. Appl. 6*, 11 (1986), 56–67.

[HRRG08] HAN C., RISSER E., RAMAMOORTHI R., GRINSPUN E.: Multiscale texture synthesis. *ACM Trans. Graph. 27*, 3 (2008), 1–8.

[ICS05] ISENBERG T., CARPENDALE M. S. T., SOUSA M. C.: Breaking the Pixel Barrier. In *Proc. of Workshop on Computational Aesthetics in Graphics, Visualization and Imaging* (2005), pp. 41–48.

[IH03] IGARASHI T., HUGHES J. F.: Smooth meshes for sketch-based freeform modeling. In *I3D '03: Proceedings of the 2003 symposium on Interactive 3D graphics* (2003), ACM, pp. 139–142.

[IMIM08] IJIRI T., MÊCH R., IGARASHI T., MILLER G.: An example-based procedural system for element arrangement. *Computer Graphics Forum 27* (2008).

[JC08] JOSHI P., CARR N.: Repoussé: Automatic inflation of 2D artwork. In *SBIM'08* (2008), pp. 49–55.

[Joh02] JOHNSTON S. F.: Lumo: illumination for cel animation. In *NPAR '02* (2002), ACM, pp. 45–52.

[KEBK05] KWATRA V., ESSA I., BOBICK A., KWATRA N.: Texture optimization for example-based synthesis. In *ACM Trans. on Graphics* (2005), vol. 24, pp. 795–802.

[LÓ1] LÉVY B.: Constrained texture mapping for polygonal meshes. In *SIGGRAPH '01* (2001), ACM, pp. 417–424.

[LH06] LEFEBVRE S., HOPPE H.: Appearance-space texture synthesis. In *ACM Trans. on Graphics* (2006), vol. 25, pp. 541–548.

[LLH04] LIU Y., LIN W.-C., HAYS J.: Near-regular texture analysis and manipulation. *ACM SIGGRAPH 2004 Papers* (2004), 368–376.

[LM99] LEUNG T., MALIK J.: Recognizing surfaces using three-dimensional textons. *Computer Vision (Proceedings of ICCV 1999)* (1999).

[NSACO05] NEALEN A., SORKINE O., ALEXA M., COHEN-OR D.: A sketch-based interface for detail-preserving mesh editing. *ACM Trans. Graph. 24*, 3 (2005), 1142–1147.

[OBW*08] ORZAN A., BOUSSEAU A., WINNEMÖLLER H., BARLA P., THOLLOT J., SALESIN D.: Diffusion curves: A vector representation for smooth-shaded images. In *ACM Trans. on Graphics* (2008), vol. 27.

[RLA*06] RITTER L., LI W., AGRAWALA M., CURLESS B., SALESIN D.: Painting with texture. In *EGSR'06* (2006).

[SCSI08] SIMAKOV D., CASPI Y., SHECHTMAN E., IRANI M.: Summarizing visual data using bidirectional similarity. In *CVPR '08* (2008).

[SKvW*92] SEGAL M., KOROBKIN C., VAN WIDENFELT R., FORAN J., HAEBERLI P.: Fast shadows and lighting effects using texture mapping. *SIGGRAPH Comput. Graph. 26*, 2 (1992), 249–252.

[SLWS07] SUN J., LIANG L., WEN F., SHUM H.-Y.: Image vectorization using optimized gradient meshes. *ACM Trans. on Graphics 26*, 3 (2007), 11.

[Wel04] WELSH T.: Parallax mapping with offset limiting: A per-pixel approximation of uneven surfaces. *Infiscape Corporation* (2004).

[Wol94] WOLBERG G.: *Digital Image Warping*. IEEE Computer Society Press, Los Alamitos, CA, USA, 1994.

[WSTS08] WU T.-P., SUN J., TANG C.-K., SHUM H.-Y.: Interactive normal reconstruction from a single image. *ACM Trans. Graph. 27*, 5 (2008), 1–9.