Title

A texture synthesis method for liquid animations

Authors

Bargteil, AW
Sin, F
Michaels, JE
et al.

Publication Date

2006-09-02

Eurographics/ ACM SIGGRAPH Symposium on Computer Animation (2006)
M.-P. Cani, J. O'Brien (Editors)

A Texture Synthesis Method for Liquid Animations

Adam W. Bargteil Funshing Sin Jonathan E. Michaels Tolga G. Goktekin James F. O'Brien

University of California, Berkeley

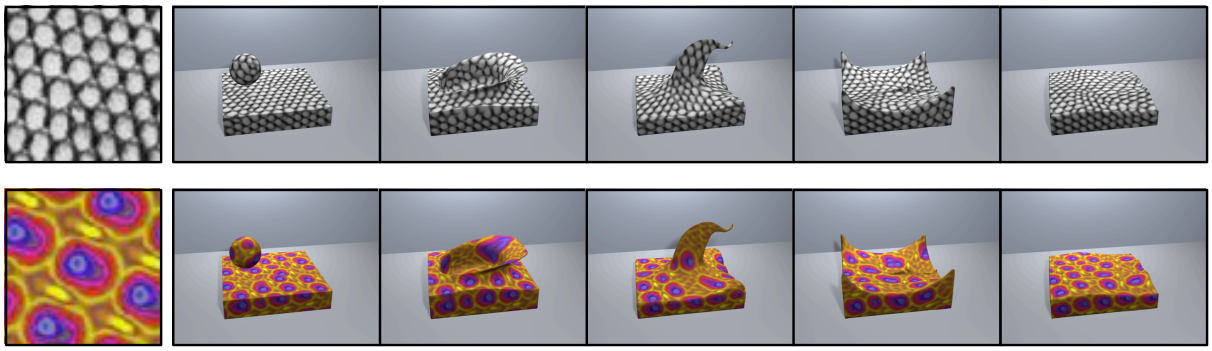

Figure 1: *This splashing motion was textured using our texture synthesis technique for liquid animations. Despite topological changes and significant surface distortions, the salient characteristics of the synthesized texture remain constant.*

Abstract

In this paper we present a method for synthesizing textures on animated liquid surfaces generated by a physically based fluid simulation system. Rather than advecting texture coordinates on the surface, our algorithm synthesizes a new texture for every frame using an optimization procedure which attempts to match the surface texture to an input sample texture. By synthesizing a new texture for every frame, our method is able to overcome the discontinuities and distortions of an advected parameterization. We achieve temporal coherence by initializing the surface texture with color values advected from the surface at the previous frame and including these colors in the energy function used during optimization.

Categories and Subject Descriptors (according to ACM CCS): I.3.5 [Computer Graphics]: Computational Geometry and Object Modeling—Surfaces and object representations; Physically based modeling; I.3.7 [Computer Graphics]: Three-Dimensional Graphics and Realism—Animation; Color, shading, shadowing, and texturing; I.6.8 [Simulation and Modeling]: Types of Simulation—Animation.

Keywords: Texture synthesis, texture mapping, surface texturing, natural phenomena, physically based animation, fluid simulation, surface tracking, surface modeling, semi-Lagrangian contouring.

1. Introduction

Liquid simulation techniques have become a standard tool in production environments, producing extremely realistic liquid motion in a variety of films, commercials, and video games. Surface texturing is an essential computer graphics tool, which gives artists additional control over their results by allowing them to stylize surfaces or add detail to low-resolution simulations. For example, an artist could use texturing techniques to add the appearance of foam to a wave, bubbles to beer, or fat globules to soup. Unfortunately, texturing liquid surfaces is difficult because the surfaces have no inherent parameterization.

Creating a temporally consistent parameterization is extremely difficult for two primary reasons. First, liquid simulations are characterized by their complex and frequent topological changes. These topological changes result in signifi-

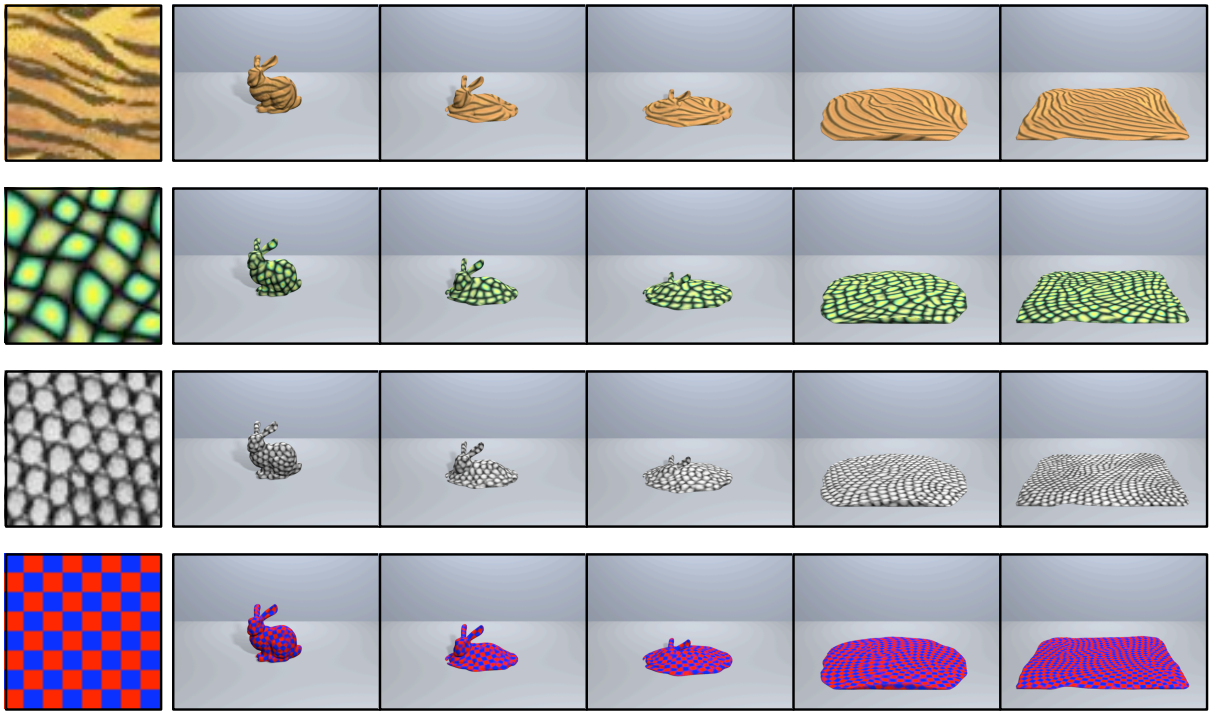

Figure 2: *This figure shows several textures applied to a simulation of a melting bunny.*

cant discontinuities in any parameter tracked on the surface. Second, liquid surfaces tend to stretch and compress dramatically over the course of a simulation. Similarly, an advected parameterization will also stretch and compress. While it may be appropriate to squash and stretch some textures with the motion of the liquid surface, many textures, such as fat globules on the surface of soup, should maintain a particular scale even as the liquid surface deforms. For these reasons, advected texture coordinates are often unsuitable for texturing liquid surfaces.

In this paper we present a method for generating textures on animated liquid surfaces. Rather than advecting texture coordinates on the surface, we synthesize a new texture for every frame. We initialize the texture with color values advected from the surface at the previous frame. We then run an optimization procedure which attempts to match the surface texture to an input sample texture and, for temporal coherence, the advected colors.

By synthesizing a new texture for every frame, our method is able to overcome the discontinuities and distortions of an advected parameterization. We avoid discontinuities in the parameterization due to topological changes by building a new parameterization of the surface for each frame. Discontinuities in advected colors are removed during the optimization procedure. Similarly, we avoid stretched and compressed parameterizations; because we optimize the surface texture for every frame, it maintains a consistent level of detail throughout the simulation. We ensure temporal coherence by initializing the optimization with the advected colors and including a coherence term in the energy function used during optimization. As a result, our method is able to produce textures with excellent temporal coherence, while still matching the input sample texture.

2. Related Work

Soon after the introduction of fluid simulation techniques to computer graphics, researchers began experimenting with texturing these simulations. The simplest approach, demonstrated by Witting [Wit99] advects texture coordinates through the flow field and uses these texture coordinates to lookup color in the texture map. Unfortunately, over time, the texture becomes progressively more distorted. To address this distortion, Stam [Sta99] advects three separate layers of texture coordinates, each of which is periodically reset. The final texture map is then a superposition of these three texture maps. Neyret [Ney03] built on this approach and also advects several layers of textures. Additionally, he computes and advects the local accumulated deformation for each texture layer. Using this deformation measure, he combines the various texture layers to arrive at a final texture, which is well adapted to the local deformation. When using procedu-

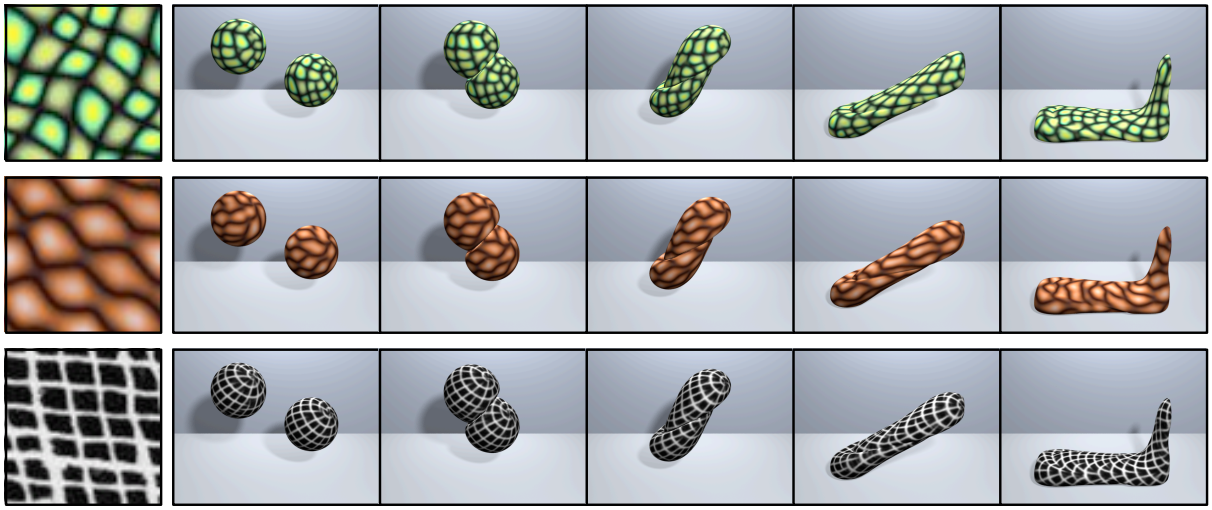

Figure 3: *In this simulation, two balls of viscoelastic fluid are thrown at each other and merge. The texturing method handles this topological change without introducing any objectionable "pops."*

ral noise-based textures, he combines the layers in frequency space to avoid ghosting effects and contrast fading.

While these techniques work relatively well for advecting textures through general fluid simulations, they are not directly applicable in the case of free-surface liquid simulation. In this case, we wish to texture the liquid surface rather than the fluid volume. To address the particular context of liquids, Rasmussen et al. [RNGF03] describes a method that advects texture particles, initialized near the free surface, through the fluid flow field. During rendering, when a ray intersects the surface the texture coordinates from the nearest 64 particles are interpolated to provide a texture coordinate for the surface point being shaded. In a similar approach, Wiebe and Houston [WH04] and Houston et al. [HNB*06] stored three-dimensional texture coordinates in a grid structure and advected them like any other scalar field. To avoid artifacts resulting from volumetric advection the authors used extrapolation techniques to force the gradient of the texture field to be perpendicular to the free surface normal. Bargteil et al. [BGOS06] introduced a free-surface tracking method that allowed for advection of texture coordinates (or other surface properties) on the actual surface. Unfortunately, all these approaches suffer from problems with discontinuous and distorted parameterizations.

Bargteil et al. [BGOS06] also proposed generating textures with a reaction-diffusion simulation [Tur91, WK91] driven by advected morphogens. Their approach is able to deal with topological changes and surface distortions. Unfortunately, their approach only admits textures which can be generated from reaction-diffusion simulations and suffers from the fact that very small perturbations of the surface can substantially change the resulting texture. Consequently, small surface motion can cause large changes in the texture.

In this paper we take an approach similar to the reaction-diffusion textures of Bargteil et al. [BGOS06] and synthesize the texture for every frame. However, we use an example-based, rather than simulation based, texture synthesis method. Example based texture synthesis has been a popular research area in computer graphics with early work being done by Heeger and Bergen [HB95] and De Bonet [Bon97]. More recently, Efros and Leung [EL99], Wei and Levoy [WL00], and Efros and Freeman [EF01] have demonstrated extremely impressive results. Our texture synthesis is based on the flexible optimization approach developed by Kwatra et al. [KEBK05].

Any surface texturing method must construct some parameterization of the surface. Numerous methods for the automatic generation of parameterizations of arbitrary surfaces exist. These methods can be roughly divided into two categories: methods that parameterize a set of (potentially overlapping) small patches and methods that attempt to find a globally-optimal parameterization. Our work belongs to the first category. The pioneering work of Bennis et al. [BVI91] introduced the idea of using piecewise parameterizations of surfaces for texture mapping. Later, Maillot et al. [MYV93] introduced the idea of texture atlases, which allow the surface to be broken up into patches where each patch has its own parameterization and texture. They also introduced a widely used surface-flattening heuristic. The lapped textures technique of Praun et al. [PFH00] places overlapping, irregularly shaped texture patches on the surface. This approach works quite well for many textures and is very similar to our approach, the primary difference being that we opti-

mize the mapping of texture patches onto the surface. Concurrently, Wei and Levoy [WL01] and Turk [Tur01] introduced texture synthesis methods which create local parameterizations of the surface and then synthesize textures directly on the surface. However, their greedy texture synthesis approach differs from the optimization approach presented here. More recently, Sorkine et al. [SCOGL02] introduced a greedy method for creating bounded-distortion local surface parameterizations based on a simple distortion metric, which was introduced by Sander et al [SSGH01].

When implementing the approach described in this paper, we recommend having Bargteil et al. [BGOS06], Kwatra et al. [KEBK05], Sorkine et al. [SCOGL02], Wei and Levoy [WL01], and Praun et al. [PFH00] on hand.

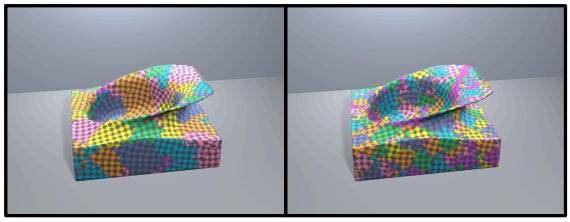

Figure 4: *These images show the patches used in our optimization process. Each patch will be assigned colors from one region of the input texture and overlapping patches will have their colors blended together. It is interesting to note that we cannot always construct perfect patches and this leads to small holes in some of the patches.*

3. Methods

Our method is built from three relatively new computer graphics technologies: the ability to track surface properties in liquid simulations [BGOS06], techniques to parameterize overlapping patches of surface [PFH00, SCOGL02], and an optimization-based technique for texture synthesis [KEBK05]. By combining these three methods we have developed a new algorithm that generates coherent, undistorted textures on liquid surfaces based on sample textures.

3.1. Surface Tracking

The motion in our examples is generated using a state-of-the-art physically based liquid simulator. More specifically, we use the staggered-grid data structure of Foster and Metaxas [FM96], the semi-Lagrangian advection method introduced by Stam [Sta99], the extrapolation boundary condition of Enright et al. [EMF02], the viscoelasticity model of Goktekin et al. [GBO04], and the surface tracking method of Bargteil et al. [BGOS06].

A necessary feature of any liquid simulation system is the ability to track the liquid's free surface. While several techniques exist, the semi-Lagrangian contouring method presented by Bargteil et al. [BGOS06] also provides a mapping between liquid surfaces at adjacent timesteps. This mapping can be used to accurately track arbitrary surface properties on the actual liquid surface at negligible additional cost. We use this feature to advect colors and parametric directions on the surface through time. If a different surface tracking method is preferred, the texture particle interpolation method developed by Rasmussen et al. [REN*04] could be used to advect colors, though this approach would introduce significant computational expense and may cause unwanted blurring between nearby surfaces.

3.2. Surface Parameterization

To apply any texture to the surface, we must construct some parameterization of the surface. For our optimization-based synthesis method (see Section 3.3), we create local parameterizations of a set of overlapping patches on the surface (see Figure 4). For each patch, the parameterization allows us to map colors on the surface to two-dimensional texture space and vice versa.

The surface meshes generated by the liquid simulation system, which uses a marching cubes method, contain many poorly shaped triangles and large dihedral angles. Unfortunately, these meshes do not admit even local parameterizations without significant distortions. Consequently, as a preprocessing step, we re-tile the surfaces using the method presented by Turk [Tur92]. This re-tiling step also allows us to control the resolution of the texture on the surface [WL00]. We then uniformly sample points on the surface using the repulsion method described by Turk [Tur92]. For each point, p_i we grow a surface patch using the method described by Sorkine et al. [SCOGL02]. The principal difference is that we allow our patches to overlap, rather than creating disjoint patches.

We grow our patches by first mapping the triangle containing p_i to texture space. This triangle is oriented based on the parametric direction advected during surface tracking (Section 3.1). We then iteratively add vertices adjacent to the patch. We place each vertex in texture space at the point which minimizes the distortion to the triangles created by adding the vertex. We reject any vertex that creates overly distorted triangles or causes any self-intersections of the patch in texture space. Finally, we apply the patch optimization procedure described by Praun et al. [PFH00]. This optimization involves solving a sparse linear system and seeks to align all of the triangles in the patch with the advected parameter directions.

3.3. Texture Synthesis

Due to discontinuities, surface stretching/compression, or blurring of the surface signal, the distorted pattern of colors generated by the semi-Lagrangian mapping typically

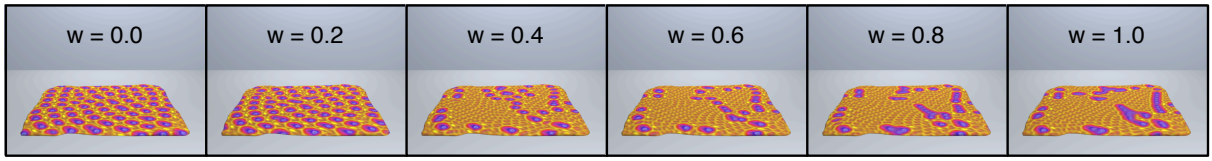

Figure 5: *This figure shows the last frame of an animation similar to those in Figure 2 for a variety of w values. For low values of w, the final texture more closely matches the input texture. As w increases there is more temporal coherence between the frames, but the optimization is less able to match the details of the sample texture.*

will not be a good match to the original texture pattern. To force the surface colors to more closely match the input sample texture, we employ an optimization-based texture synthesis method based on the one presented by Kwatra et al. [KEBK05].

Throughout the optimization we store three types of color for each vertex of each surface patch: the *current colors*, the *advected colors*, and the *best-match colors*. The *current colors* refer to the colors currently stored on the mesh—a blend of all the best-match colors of all the patches overlapping a given vertex. The current colors represent the current state of the optimization and change with each optimization step. When optimization is complete the current colors will define the final texture on the surface. The *advected colors* refer to the distorted colors generated through the semi-Lagrangian mapping. The advected colors are used to initialize the current colors, but remain constant during optimization. The *best-match colors* are the colors, chosen from the input texture sample, that most closely match the current colors in a given patch.

Each iteration of the optimization process comprises four steps for each patch:

1. Map the current and advected colors from the surface to texture space.
2. Find the best-match to the current colors and advected colors in the sample texture.
3. Map these best-match colors to the surface.
4. Update the current colors on the surface.

Step (1) uses the parameterization described in Section 3.2 to map the current and advected colors from the surface to texture space. Step (2) finds the best match in the input sample texture to both the advected and current colors by finding the region in the input sample texture which minimizes the energy function

$$E(\boldsymbol{c},\boldsymbol{a},\boldsymbol{b}) = \sum_{i,j} g(i,j) \left\| (1-w)(c_{ij} - b_{ij}) + w(a_{ij} - b_{ij}) \right\|^2,$$

where $\boldsymbol{c}$ are the current colors for the patch (mapped to texture space), $\boldsymbol{a}$ are the advected colors, $\boldsymbol{b}$ are the best-match colors (which is the variable we are minimizing over), i and j vary over the two-dimensional texture region, $g(\cdot)$ is a Gaussian weighting function which ensures that colors near the center of the patch have more weight, and w is a weighting parameter that trades off temporal coherence and matching the sample texture (see Figure 5). Step (3) maps these best-match colors from texture space to the surface mesh. Finally, step (4), removes the contribution of the previous best-match colors from the current colors stored at the mesh vertices and blends in the colors of the new best match. Following Kwatra et al. [KEBK05], we perform the optimization at several mesh resolutions and for several patch sizes at each resolution.

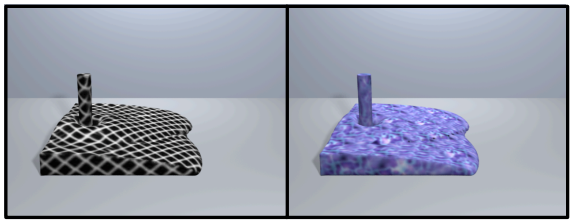

Figure 6: *This figure shows two textures applied to an animation of a tank filling with viscous fluid.*

This optimization approach is particularly appealing in our context. In many parts of the surface that have experienced minimal distortion, the advected colors may quite closely match the sample texture. Consequently, the optimization makes only minor changes. Additionally, we achieve temporal coherence by initializing the optimization with the advected colors and including a term in the energy function which attempts to match these advected colors. This temporal coherence is demonstrated in Figure 3.

4. Results

We have implemented the method described in this paper and demonstrated it with a variety of fluid motions and texture samples. The fluid motions demonstrate significant squashing and stretching of the surface as well as a variety of topological changes. Our method generates surface textures which match the input sample texture while remaining temporally coherent.

Figure 1 shows an animation of a splash created when a ball of fluid is thrown into a shallow pool of fluid. The re-

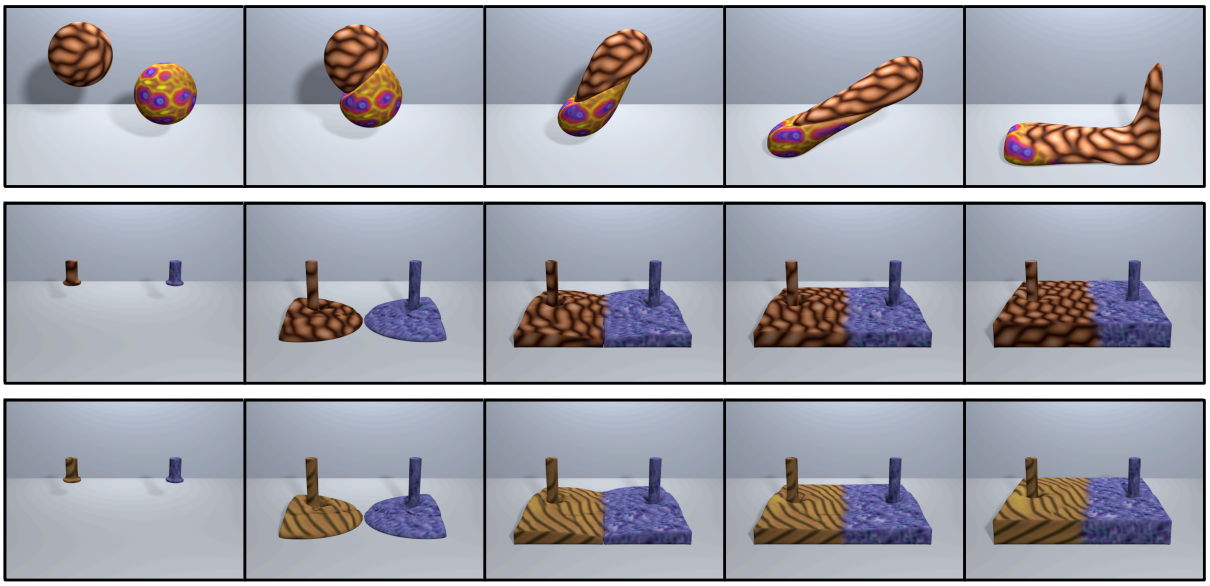

Figure 7: *This figure shows some examples with multiple textures. After initialization the optimization searched both textures for the best match.*

sulting motion demonstrates significant stretching of the surface, but the surface texture does not become overly distorted and always provides a good match to the input sample texture. If we simply advected colors on the surface, the texture would significantly blur and distort.

Figure 2 shows an animation of a melting bunny with a checkerboard texture. Though the resulting texture is not a perfect checkerboard, which is impossible because the surface is not developable, locally the texture quite closely matches the checkerboard sample, and globally the texture does resemble a checkerboard.

Figure 3 shows an animation of two balls of fluid thrown at each other. Because we explicitly include temporal coherence in the energy function there is no noticeable "pop" when the two spheres merge, rather they gradually move toward a continuous texture which matches the sample. Figure 6 shows an additional example of viscous fluid filling a tank with two different textures. Figure 7 shows some examples with multiple textures.

Unfortunately, our implementation is not particularly fast. Re-tiling a surface mesh takes about one minute, generating and optimizing the surface patches takes between fifteen and thirty minutes for a single frame, and the texture optimization takes between one and fifteen minutes per frame, depending on the amount of distortion of the texture and the number of vertices in the mesh. Fortunately, the re-tiling and patch generation can be done in parallel, so they do not create a significant bottleneck in a traditional rendering pipeline. Additionally, in this work we were more concerned with developing a method which produces high quality results rather than one optimized for speed. We believe the general method could be made much faster, perhaps using ideas developed by Magda and Kriegman [MK03].

Concurrently with our work, Kwatra et al. [KAK*06a, KAK*06b] have developed a very similar example-based texture synthesis method for fluids. However, they do not build and optimize patches as a precomputation, but rather construct color neighborhoods on the fly using the method presented by Turk [Tur01]. They also have a more developed texture synthesis module which uses K-means trees to find the best match in the example texture rather than our brute force approach.

5. Conclusion

We have presented a new technique for texturing liquid surfaces, which overcomes the discontinuities and distortions of an advected parameterization while maintaining excellent temporal coherence. Our method is able to handle a wide variety of input sample textures and should prove to be a useful tool for artists, complementing existing texturing techniques such as procedural texturing [EMP*02], and advected texture maps.

6. Acknowledgments

We thank the other members of the Berkeley Graphics Group, the anonymous reviewers and Vivek Kwatra for their helpful criticism and comments and Greg Turk for his mesh

retiling code. This work was supported in part by California MICRO 04-066 and 05-044, and by generous support from Apple Computer, Pixar Animation Studios, Autodesk, Intel Corporation, Sony Computer Entertainment America, and the Alfred P. Sloan Foundation. Adam Bargteil was supported in part by a Siebel Scholarship.

References

[BGOS06] BARGTEIL A. W., GOKTEKIN T. G., O'BRIEN J. F., STRAIN J. A.: A semi-Lagrangian contouring method for fluid simulation. *ACM Trans. Graph. 25*, 1 (2006).

[Bon97] BONET J. S. D.: Multiresolution sampling procedure for analysis and synthesis of texture images. In *the Proceedings of ACM SIGGRAPH 1997* (1997), pp. 361–368.

[BVI91] BENNIS C., VÉZIEN J.-M., IGLÉSIAS G.: Piecewise surface flattening for non-distorted texture mapping. In *the Proceedings of ACM SIGGRAPH 1991* (1991), pp. 237–246.

[EF01] EFROS A. A., FREEMAN W. T.: Image quilting for texture synthesis and transfer. In *the Proceedings of ACM SIGGRAPH 2001* (2001), pp. 341–346.

[EL99] EFROS A. A., LEUNG T. K.: Texture synthesis by non-parametric sampling. In *Proceedings of the International Conference on Computer Vision-Volume 2* (September 1999), pp. 1033–1038.

[EMF02] ENRIGHT D. P., MARSCHNER S. R., FEDKIW R. P.: Animation and rendering of complex water surfaces. In *the Proceedings of ACM SIGGRAPH 2002* (July 2002), pp. 736–744.

[EMP*02] EBERT D. S., MUSGRAVE K. F., PEACHEY D., PERLIN K., WORLEY S.: *Texturing & Modeling: A Procedural Approach, Third Edition*. Morgan Kaufmann, December 2002.

[FM96] FOSTER N., METAXAS D.: Realistic animation of liquids. In *Graphics Interface 1996* (May 1996), pp. 204–212.

[GBO04] GOKTEKIN T. G., BARGTEIL A. W., O'BRIEN J. F.: A method for animating viscoelastic fluids. In *Proceedings of ACM SIGGRAPH 2004* (Aug. 2004), pp. 463–468.

[HB95] HEEGER D. J., BERGEN J. R.: Pyramid-based texture analysis/synthesis. In *the Proceedings of ACM SIGGRAPH 1995* (1995), pp. 229–238.

[HNB*06] HOUSTON B., NIELSEN M. B., BATTY C., NILSSON O., MUSETH K.: Hierarchical RLE level set: A compact and versatile deformable surface representation. *ACM Trans. Graph. 25*, 1 (2006), 151–175.

[KAK*06a] KWATRA V., ADALSTEINSSON D., KWATRA N., CARLSON M., LIN M.: *Texturing Fluids*. Tech. rep., University of North Carolina at Chapel Hill, 2006.

[KAK*06b] KWATRA V., ADALSTEINSSON D., KWATRA N., CARLSON M., LIN M.: Texturing fluids. In *the Proceeding of ACM SIGGRAPH 2006 Sketches & Applications* (2006).

[KEBK05] KWATRA V., ESSA I., BOBICK A., KWATRA N.: Texture optimization for example-based synthesis. *ACM Trans. Graph. 24*, 3 (2005), 795–802.

[MK03] MAGDA S., KRIEGMAN D.: Fast texture synthesis on arbitrary meshes. In *Proceedings of the 14th Eurographics workshop on Rendering* (2003), pp. 82–89.

[MYV93] MAILLOT J., YAHIA H., VERROUST A.: Interactive texture mapping. In *the Proceedings of ACM SIGGRAPH 1993* (1993), pp. 27–34.

[Ney03] NEYRET F.: Advected textures. In *ACM SIGGRAPH/Eurographics symposium on Computer animation* (2003), pp. 147–153.

[PFH00] PRAUN E., FINKELSTEIN A., HOPPE H.: Lapped textures. In *the Proceeding of ACM SIGGRAPH 2000* (2000), pp. 465–470.

[REN*04] RASMUSSEN N., ENRIGHT D., NGUYEN D., MARINO S., SUMNER N., GEIGER W., HOON S., FEDKIW R.: Directable photorealistic liquids. In *ACM SIGGRAPH/Eurographics symposium on Computer animation* (2004), pp. 193–202.

[RNGF03] RASMUSSEN N., NGUYEN D. Q., GEIGER W., FEDKIW R. P.: Smoke simulation for large-scale phenomena. In *the Proceedings of ACM SIGGRAPH 2003* (July 2003), pp. 703–707.

[SCOGL02] SORKINE O., COHEN-OR D., GOLDENTHAL R., LISCHINSKI D.: Bounded-distortion piecewise mesh parameterization. In *the Proceedings of IEEE Visualization '02* (2002), pp. 355–362.

[SSGH01] SANDER P. V., SNYDER J., GORTLER S. J., HOPPE H.: Texture mapping progressive meshes. In *the Proceedings of ACM SIGGRAPH 2001* (2001), pp. 409–416.

[Sta99] STAM J.: Stable fluids. In *the Proceedings of ACM SIGGRAPH 99* (Aug. 1999), pp. 121–128.

[Tur91] TURK G.: Generating textures on arbitrary surfaces using reaction-diffusion. In *the Proceedings of ACM SIGGRAPH 1991* (1991), pp. 289–298.

[Tur92] TURK G.: Re-tiling polygonal surfaces. In *the Proceedings of ACM SIGGRAPH 1992* (1992), pp. 55–64.

[Tur01] TURK G.: Texture synthesis on surfaces. In *the Proceedings of ACM SIGGRAPH 2001* (2001), pp. 347–354.

[WH04] WIEBE M., HOUSTON B.: The tar monster: Creating a character with fluid simulation. In *the Proceedings of ACM SIGGRAPH 2004 Sketches & Applications* (2004).

[Wit99] WITTING P.: Computational fluid dynamics in a traditional animation environment. In *the Proceedings of ACM SIGGRAPH 1999* (1999), pp. 129–136.

[WK91] WITKIN A., KASS M.: Reaction-diffusion textures. In *the Proceedings of ACM SIGGRAPH 1991* (1991), pp. 299–308.

[WL00] WEI L.-Y., LEVOY M.: Fast texture synthesis using tree-structured vector quantization. In *the Proceedings of ACM SIGGRAPH 2000* (2000), pp. 479–488.

[WL01] WEI L.-Y., LEVOY M.: Texture synthesis over arbitrary manifold surfaces. In *the Proceedings of ACM SIGGRAPH 2001* (2001), pp. 355–360.