



Distribution Update of Deformable Patches for Texture Synthesis on the Free Surface of Fluids

Jonathan Gagnon¹, Julián E. Guzmán², Valentin Vervondel², François Dagenais², David Mould³, and Eric Paquette²

¹FOLKS VFX

²École de technologie supérieure, Montreal, Canada

³Carleton University, Ottawa, Canada

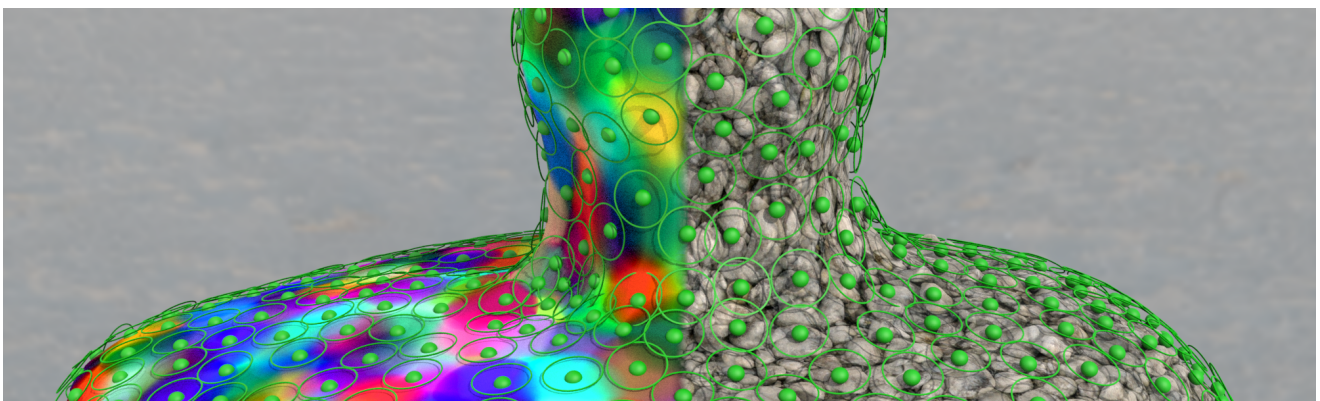


Figure 1: Texturing fluid animation with deformable patches distributed as a Poisson disk distribution. We adapt the Poisson disk distribution by fading in and fading out patches associated with newly added and removed Poisson disks. Deformable patches and their resulting blending are shown on the left side and the texture mapping appears on the right.

Abstract

We propose an approach for temporally coherent patch-based texture synthesis on the free surface of fluids. Our approach is applied as a post-process, using the surface and velocity field from any fluid simulator. We apply the texture from the exemplar through multiple local mesh patches fitted to the surface and mapped to the exemplar. Our patches are constructed from the fluid free surface by taking a subsection of the free surface mesh. As such, they are initially very well adapted to the fluid's surface, and can later deform according to the free surface velocity field, allowing a greater ability to represent surface motion than rigid or 2D grid-based patches. From one frame to the next, the patch centers and surrounding patch vertices are advected according to the velocity field. We seek to maintain a Poisson disk distribution of patches, and following advection, the Poisson disk criterion determines where to add new patches and which patches should be flagged for removal. The removal considers the local number of patches: in regions containing too many patches, we accelerate the temporal removal. This reduces the number of patches while still meeting the Poisson disk criterion. Reducing areas with too many patches speeds up the computation and avoids patch-blending artifacts. The final step of our approach creates the overall texture in an atlas where each texel is computed from the patches using a contrast-preserving blending function. Our tests show that the approach works well on free surfaces undergoing significant deformation and topological changes. Furthermore, we show that our approach provides good results for many fluid simulation scenarios, and with many texture exemplars. We also confirm that the optical flow from the resulting texture matches the fluid velocity field. Overall, our approach compares favorably against recent work in this area.

CCS Concepts

• **Computing methodologies** → Texturing;

1. Introduction

Texture mapping is commonly used to add details to 3D surfaces. However, the texture may be distorted by surface curvature. Dynamic meshes introduce further complexity, and fluid animations are especially challenging, since the surface topology can change. Texturing a dynamic mesh is still a nontrivial problem with unanswered questions.

One category of methods applicable to fluids synthesizes the texture with a patch-based search [KAK*07]. This is very effective in maintaining the overall look of the texture exemplar. Nevertheless, it makes it much harder for the pattern to be deformed according to the flow of the fluid.

Another category of methods synthesizes the texture by advecting patches on the surface. This greatly improves the temporal coherence of the synthesized texture, which is why we decided to focus our attention on such methods. Some methods rely on “rigid” patches [GDP16] and have a tendency to show “blocky” artifacts, where the resulting animation feels too rigid since it does not completely respect the velocity field.

Other methods use 2D deformable grids [YNBH11], and while this can be effective for deformations on mostly 2D fluids, it is not directly applicable to 3D fluids with splashes and topological changes. Small details such as splashes are also hard to handle, as it is difficult to have a good distribution of patches on elements with small surface areas. It is hard for the Poisson disk distribution techniques used by most patch-based methods to balance between the number of disks to assign to small regions and patch distortion concerns when wrapping the patches onto the fluid surface. Ultimately, concerns related to the deformation and distribution of patches necessitate the use of small texture patches, which limits the range of applicability of earlier patch-based texture synthesis methods. Furthermore, patch-based methods face difficulties in handling converging fluid flows. While some methods provide reasonable results for 2D flows, this problem is exacerbated when considering 3D fluids, as they are prone to accumulating many patches in some areas depending on the flow and surface curvature. The source of the problem is that patches are faded at a constant rate, causing an accumulation of fading patches in regions where the fluid converges or 3D fluids merge together. Patch-based methods lead to artifacts when the density of patches is too high.

Methods relying on layered patches [GDP16] suffer from a degradation of the synthesized texture which is composed of many small inconsistent pieces of the exemplar. Other methods that rely on contrast-preserving blending [YNBH11] are hindered by greatly reduced blending quality, as the contrast-preserving blending can lead to colors outside the $[0, 1]$ range, and the likelihood of this increases with the number of blended patches.

In contrast to previous patch-based methods, our patches are created directly on the surface of the fluid, by taking a subsection of the surface mesh. This ensures that the patches are not distorted at the time of creation; they can later deform according to the velocity field, providing greater ability to represent surface motion than rigid or 2D grid-based patches. We validated this advantage of our approach by comparing the texture’s optical flow with the fluid’s velocity field. To prevent patch accumulation during advection, we

propose a temporal patch removal approach that considers the local number of patches: in regions containing too many patches, we accelerate the temporal removal. The opposite problem of patch accumulation consists of a lack of patches on small details, such as splashes. We improve the distribution of patches by refining the Poisson disk criterion, making it better adapted to the context of small details and patches that need to be wrapped to the fluid’s surface. To this end, we use the normal of the surface and an ellipsoid form instead of a sampling sphere. The main contributions of the proposed texture synthesis method can be summarized as follows:

- Distortion reduction by creating patches from the surface;
- Dynamic fade-in and fade-out with adaptive speed related to density;
- Poisson disk criterion adapted to small features and splashes.

We obtain best results for high-frequency, stochastic, isotropic textures, as is characteristic of lapped texture synthesis. However, we do not make any assumptions with respect to the exemplar, and the results are reasonable for more structured textures. We tested our approach on many scenarios, and show that the resulting textures are temporally coherent and well adapted to the flow of the fluid.

2. Related work

Two texture synthesis strategies are predominantly used for texturing fluids: pixel-based and patch-based [BZ17, WLKT09]. Texturing fluids is difficult, and consequently, many methods [KEBK05, JFA*15, YNBH11] concentrate on the sub-problem of 2D fluids and flows. The texture optimization method [KEBK05] synthesizes its results in image space, making it hard to extend to the surface of 3D fluids. While the method uses patches for the synthesis, the patches are at fixed positions, and cannot deform, which makes it harder to adapt to various types of flows. LazyFluids [JFA*15] uses per-pixel best-match searches instead of patch-based searches. While this increases the precision, the patterns remain stiff as the search windows do not deform. Furthermore, it is also limited to synthesis in image space, and as such, is hard to extend to 3D fluids. The pioneering work of Stora et al. [SAC*99] advected lava clinker with simple texture from noise functions. Lagrangian texture advection [YNBH11] uses deformable patches which are advected based on the flow field. Even though this method works only in 2D, it investigated the problem of distortion of the texture pattern. Our approach extends the 2D deformable patches to deformable surface patches, allowing us to conduct texture synthesis on the free surface of arbitrary fluid simulations, and not only on 2D surfaces.

Few methods deal with 3D fluids. Kwatra et al. [KAK*07] extend the 2D texture optimization method [KEBK05] to 3D. They store the colors on the vertices of the fluid’s free surface instead of in image space. While their method can texture 3D fluids, it has the disadvantage of locking the texture resolution to the resolution of the simulation mesh. Furthermore, by still relying on rigid patches for their best match search and synthesis, they need to resample the vertices to a square grid where they conduct the texture synthesis, and then need to resample in order to transfer the new colors back to the vertices. Narain et al. [NKL*07] and Bargteil et al. [BSM*06] essentially extend the work of Kwatra et al. [KAK*07] to allow the use of a feature map, but their texture synthesis methods rely on

the same basis as Kwatra et al., and consequently inherit the same drawbacks. Similar to Lagrangian texture advection [YNBH11], the dynamic lapped texture method [GDP16] uses patches on the surface of the liquid. This method works in 3D and provides reasonable results for scenarios including splashes. However, it has two main drawbacks. First, the patches do not deform with respect to the flow of the 3D liquid, and instead, only deform to conform to the surface of the fluid. Second, the method uses layered patches, as opposed to a contrast-preserving blending. The layering has the disadvantage of showing many small and inconsistent pieces of the exemplar in regions where many patches accumulate because of the temporal removal.

Our goal is to take advantage of the best strategies from patch-based texture synthesis methods. We move away from 2D synthesis methods, and while our approach can still handle 2D fluids, it is built for 3D fluids. We improve upon the deformable patches of Lagrangian texture advection [YNBH11], while also using the 3D patch advection of dynamic lapped textures [GDP16]. Given our new deformable patches, our texture deformation better follows the fluid flow than the non-deforming patch-based search methods [BSM*06, KAK*07, NKL*07].

3. Overview

In this section, we present an overview of the proposed approach. Key concepts are illustrated in Fig. 2.

We start with an animated mesh, where each vertex has a velocity vector, usually arising from a fluid simulation. The goal is to use patch-based synthesis to synthesize a texture over the mesh. The first step of the approach is to sample the surface with a Poisson disk distribution on the first frame of the animation. Next, we create a *deformable patch* which contains uv coordinates for each Poisson disk based on the mesh vertices surrounding it. For each patch and each patch vertex, we compute a contribution weight that will be used to compute the texel's color. Once the weights have been computed, we can finalize the texture synthesis by calculating, for each texel, a final color, which is a weighted combination of the contributing patches.

In subsequent frames of the animation, we advect the Poisson disks and the vertices of the deformable patches according to the velocity field. To minimize distortion, we repair the overall texture by updating the patch representation: we remove patches with excessive distortion, eliminate patches in areas that are too crowded, and add new patches where there are gaps on the surface. The new patch distribution is used to compute the texture for the current frame. Patch removal and addition are not instantaneous. Rather, patches fade out (or in) over time at a rate determined by the local patch density. Details are provided in Sec. 6.3.

Our approach is inspired by the method of Yu et al. [YNBH11], which we extend to work on free surfaces. The differences between our approach and that of Yu et al. [YNBH11] are listed in Table 1.

A Poisson disk distribution is not designed to handle details smaller than the Poisson disk radius. Therefore, we propose a solution using the plane of the surface's normal, combined with an ellipsoid volume per patch as the Poisson disk criterion, which is

Table 1: Comparison between our approach and that of Yu et al. [YNBH11].

Yu et al. [YNBH11]	Our approach
Poisson Disk sampling in 2D	Poisson Disk sampling in 3D
2D grid of vertices	3D set of vertices
Orthogonal uv projection	uv flattening [LPRM02]
2D advection	3D advection on free surface
No topological changes	Handling topological changes
Linear patch fading	Dynamic patch fading

discussed in Sec. 4. Creating deformable patches is tricky when we have a curved surface and complex mesh topology. The creation of deformable patches is discussed in Sec. 5. Deformable patches need to be updated over time in a consistent fashion in order to preserve their texture exemplar's features spatially and temporally. We fade in and out patches at a rate influenced by density of patches; details of the entire approach are discussed in Sec. 6.

4. Distribution on curved surfaces

This section describes our approach to surface sampling. We want to avoid two problematic conditions: areas of the surface with no patches, and areas on the surface with too many patches. We sample the surface with a Poisson disk distribution on the first frame. On subsequent frames, we compute a new Poisson disk distribution using the advected disks as a starting point. This allows us to check if it is possible to add new patches or if we need to delete some of them.

There are multiple Poisson disk sampling methods available; some of these work in 2D, and others in 3D. For our purpose, we need to sample curved surfaces, and we want to be able to distribute more samples in high curvature areas, such as splashes or thin threads. These concerns are important since we do not want our patches to deform too much when wrapping on the fluid surface, as this would cause unwanted texture stretch. As shown in Fig. 3a, when dealing with nearby surfaces, and only considering a 3D Poisson radius r , samples on one surface count toward the Poisson criterion of the other surface, leading to an undesired undersampling.

To fix this issue and provide an appropriate surface sampling, we propose using two criteria:

- no other Poisson disk should be inserted within the radius r of another disk only if both disks are in the same plane;
- no other Poisson disk should be inserted inside an ellipsoid defined by the surface and disk.

The plane of a specific Poisson disk is computed from the normal \mathbf{N} evaluated where the disk lies on the surface. When testing to add a new disk, we reject it only if it is in the same plane *and* if it is within the radius r . We thus test if the normal \mathbf{N}_c at the candidate location is too close to the normal \mathbf{N} of the disk, determined by checking whether the value of the dot product is above a given threshold: $\mathbf{N}_c \cdot \mathbf{N} > pa$. For results shown in this paper, we set pa to 0.5.

These criteria are illustrated in Fig. 3. We can see in Fig. 3a that

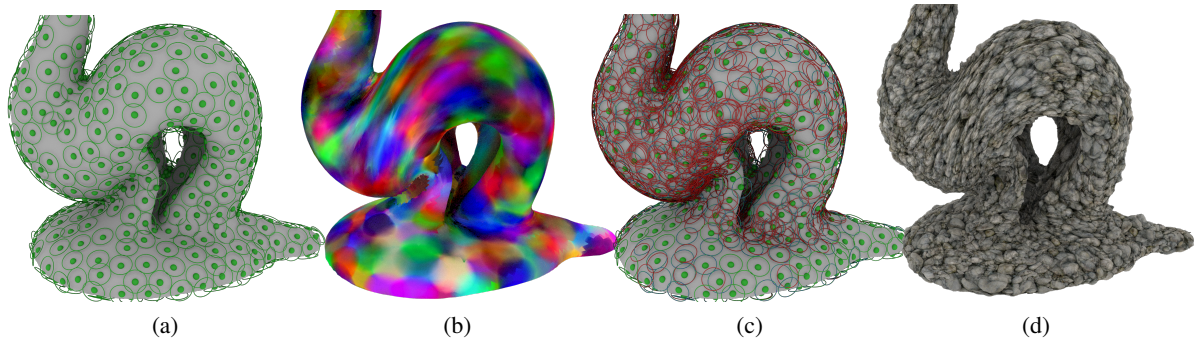


Figure 2: Key concepts of the proposed approach: (a) Initial Poisson disk distribution. (b) Visualization of deformable patches, where each patch is assigned a different color. (c) Patch distribution update; green disks are kept, red disks will be removed, blue disks are added. (d) The final synthesized texture.

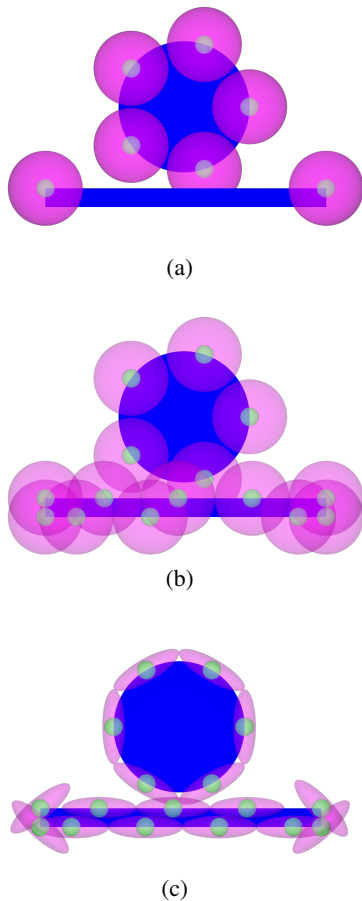


Figure 3: Example of underlying surfaces (in blue), with Poisson disks (in magenta) and their related Poisson points (in green): (a) Poisson disks with 3D radius r criterion. (b) Poisson disks with distance and plane criteria. (c) Poisson disks with ellipsoid distance and plane criteria.

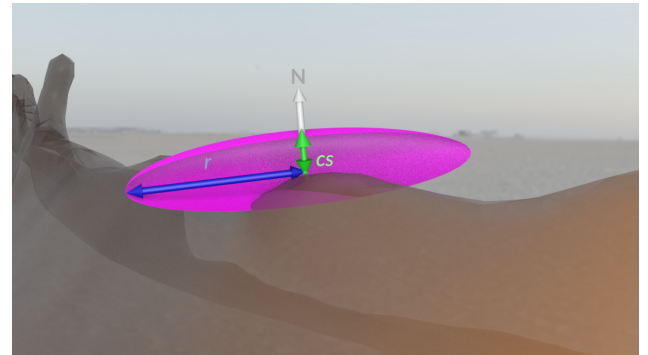


Figure 4: Our k -criterion corresponding to an ellipsoid (shown in magenta) having thickness cs in the normal direction \mathbf{N} and width r . Candidate samples within this ellipsoid are rejected.

there are regions not covered by any sample when only the distance r is used. In Fig. 3b, we add the orientation to the criterion, but some regions remain uncovered. Finally, Fig. 3c shows the result of using the ellipsoid in combination with the orientation, producing a better surface sampling.

We will refer to the ellipsoid criterion as the k -criterion. A candidate disk located at \mathbf{p} is too close to another one if the following condition holds:

$$k(\mathbf{p}) = \frac{x^2}{r^2} + \frac{y^2}{r^2} + \frac{z^2}{cs^2} < 1, \quad (1)$$

where cs is the ellipsoid thickness and x , y , and z are the coordinates of the tested point in the tangent space of the Poisson disk, where z points in the same direction as the normal \mathbf{N} . The geometry is illustrated in Fig. 4. This ellipsoid thickness should be smaller than the smallest detail of the fluid animation. The point (x, y, z) is inside the ellipsoid when $k(\mathbf{p})$ is less than 1. With the k -criterion, we are able to fit thin surface details, as well as to address the undersampling problem shown in Fig. 5a; see Fig. 5b for an illustration of the result.

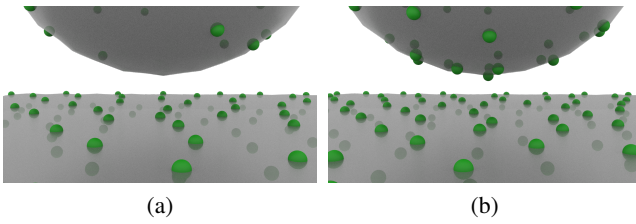


Figure 5: (a) Regular 3D Poisson disk distribution does not distribute enough disks in the region where the lower part of the sphere is close to the plane; (b) samples cover the lower part of the sphere when using our k -criterion.

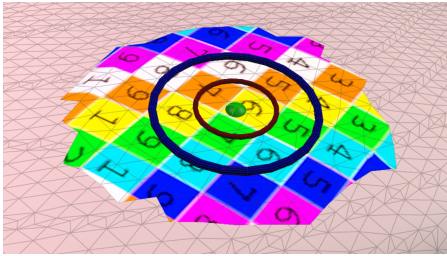


Figure 6: Representation of a deformable patch with the Poisson disk (outer ring), the selected polygons, and the uv coordinates. The kill zone (inner ring) is the threshold where we identify patches too close to each other.

5. Deformable patches

This section describes the proposed approach to handle surface deformation by using deformable patches. A deformable patch consists of the surface polygons surrounding the patch center; the polygon's vertices can then be advected independently while maintaining their texture coordinates, allowing the patch to deform.

5.1. Patch creation using surrounding polygons

A deformable patch is created by duplicating the polygons of the surface mesh surrounding the Poisson disk sample. The distance used to select polygons is slightly greater than the ellipsoid used in equation 1 to ensure that there will be enough polygons on the border of the patch (Fig 6). We exclude polygons that are not in the same plane as the Poisson disk, using the dot product test described in Sec. 4. The resulting set of polygons can be flattened to the uv coordinate space of the exemplar. To ensure minimal distortion of the uv coordinate assignment over the 3D patch, we use the parameterization given by Least Squares Conformal Maps [LPRM02].

5.2. Patch advection

We advect the vertices of each patch according to the velocity field on the surface. The examples presented in this paper relied on an implicit surface reconstruction (with independent meshes at each frame), but our method would work as well with an explicit mesh [DGP17]. Since the surface can undergo drastic and sudden

topological changes, such as splits and merges, we must update vertices accordingly and remove any part of the patch that cannot follow the surface. Using an Euclidean distance threshold mpt , we remove from the patch all vertices that end up too far from the surface of the fluid. This removal allows for the introduction of new patches, leading to an overall patch distribution that is more faithful to the motion of the fluid's surface. Vertices below this threshold are projected onto the surface. The value of the threshold mpt is adjusted considering the scene scale and the fluid simulation. If a patch ends up stretched between two disconnected components of the fluid, it is identified as distorted and removed, as explained in the next section.

5.3. Distortion estimation

Over the animation, every patch can deform, and the deformation can produce visible distortions in the resulting synthesis. Therefore, as in the work of Yu et al. [YNBH11], we detect distortion with the distortion metric of Sorkine et al. [SCOGL02]. Excessively distorted patches will be removed and replaced with undistorted ones; we consider a patch to be excessively distorted when the distortion value δ_{max} exceeds 3.

5.4. Poisson disk distribution update

With the creation and subsequent advection of deformable patches, we obtain deformed and displaced patches. We previously detected distorted patches, as described in Sec. 5.3. To maintain a uniform distribution of patches, we use the new patch positions to flag the patches to remove and insert new patches where there is a gap. In addition to distorted patches, we also want to remove patches that are too close to each other. These are the patches whose centers fall within the “kill distance” $(1 - \alpha)d$ of another patch, as illustrated in Fig. 6. Where there is a gap on the surface, we insert a new Poisson disk and its corresponding patch.

The Poisson disk distribution is done frame by frame, allowing us both to flag patches that are too close to each other, and to identify regions where new patches are needed. Finally, the distortion metric points out which patches need to be deleted because they are too distorted. The Poisson disks have four states:

- fading in
- active
- fading out (too distorted)
- fading out (too close)

The transition between states is illustrated in the accompanying video. Note that during the Poisson disk distribution update, we ignore the fading out patches.

6. Blending

The last step of the approach is to blend all patches together to synthesize the final texture atlas. This process is done on every frame of the fluid animation. First, for each vertex of each patch, a vertex weight is computed. Using this value, we then calculate the color of each texel using the blending function described in Sec. 6.4.

6.1. Patch update

After the advection, with the updated distribution, we can compute the quality of each vertex of a deformable patch at a time t . To do so, we compute a weight per vertex. A vertex weight $w_V(t)$ calculation was introduced by Yu et al. [YNBH11]:

$$w_V(t) = K_S(V)K_t(t). \quad (2)$$

The overall weight w_V is the product of a spatial weight $K_S(V)$ and a temporal weight $K_t(t)$. The weight w_V will be used in the blending computation to synthesize the texture, as described in Sec. 6.

6.2. Spatial component

The spatial component K_S improves the continuity at the borders of the patches. The idea is to get the full color of the texture at the center of the patch, and to let this color contribution fall off towards the borders. To this end, we have K_S equal to 1 at the center and 0 at the border. We use a linear blending, where K_S varies according to the distance between the vertex V and the patch center \mathbf{p} . To compute the 3D distance between V and \mathbf{p} , we modify the K_S computation given by Yu et al. [YNBH11]: instead of computing the distance between vertex V and the Poisson disk center \mathbf{p} in three dimensional space, we compute the distance between vertex V_{uv} and the disk center p_{uv} in uv space. Thus, when the patch is deformed, we always have the same distance in texture space, and the linear blending is consistent over time. Moreover, to give the user more control over the look of the texture synthesis, we also allow scaling s of the uv coordinates. The equation from Yu et al. is as follows:

$$K_S(V) = \left(1 - \frac{\|V - p\|}{d}\right) d_V Q_V. \quad (3)$$

We replace this with the following:

$$K_S(V) = \left(1 - \frac{\|V_{uv} - p_{uv}\|}{d_{uv}}\right) s d_V Q_V. \quad (4)$$

6.3. Density-based temporal component

We also modify the temporal component $K_t(t)$ from Yu et al. [YNBH11], using a density-based fading. Fading out and fading in patches over a fixed number of frames works well when the velocity field does not change too rapidly. However, in fluid simulations, the velocity on the surface can change quickly, resulting in large amounts of patches that get stacked and have to be deleted. We propose to change the fading according to the density of patches in the patch's ellipsoid volume. Yu et al. fade over a fixed number of frames τ :

$$K_t(t) = \begin{cases} \frac{t}{\tau} & \text{if } t < \tau \\ 1 & \text{if } \tau < t < t_k \\ 1 - \frac{t-t_k}{\tau} & \text{if } t_k < t < t_k + \tau \end{cases}. \quad (5)$$

In contrast to the fixed linear evolution of $K_t(t)$ in the equation above, our fading value is adjusted dynamically based on the density:

$$K_t(t) = \begin{cases} \min\left(1, \frac{t}{\tau}(\rho + 1)\right) & \text{if } t < \tau \\ 1 & \text{if } \tau < t < t_k \\ \max\left(0, 1 - \frac{t-t_k}{\tau}(\rho + 1)\right) & \text{if } t_k < t < t_k + \tau \end{cases}, \quad (6)$$

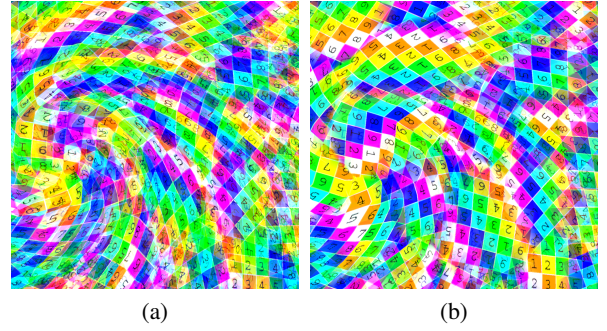


Figure 7: (a) Linear fading vs. (b) density-based fading.

where t_k is the time at which the patch is flagged to be removed, and ρ is the density of the current patch's Poisson disk. When $K_t(t)$ reaches 1 (fading in) or 0 (fading out), we terminate the fading process, thus dynamically changing the number of frames over which we apply fading. The density ρ is computed using the number of neighbors of the Poisson disk in the same plane and inside the radius r . We consider all patches when computing the density, irrespective of their current state. Patches will both fade in and fade out faster when there are more overlapping patches. Locations where we have large distortions trigger a lot of distorted patch removal, and the last line of Eq. 6 ensures their quick removal. However, removing many patches in turn triggers their replacement by many new patches. If patches here were to fade in slowly, there would be a good chance that they would be highly distorted by the time they have completely arrived, and thus they would need to be removed immediately. The first line of Eq. 6 ensures that patches fade in quickly, increasing the time during which the surface is covered by undistorted patches.

Under Equation 6, it is possible to encounter very large ρ , potentially producing a negative $K_t(t)$. When K_t becomes negative for any patch, we remove the patch immediately.

In practice, the number of overlapping patches is related to the local velocity and deformation. Changing the rate of the fading at these locations looks visually plausible and involves fewer blending artifacts, as shown in Fig. 7. Indeed, we can see in this example that the numbers from the texture exemplar are easier to read because they involve less blending. In the end, we have fewer stacking artifacts, and the final texture can be computed more quickly as it involves blending fewer patches.

6.4. Blending function

The vertex weight $w_V(t)$ is then used in the following blending function introduced by Yu et al. [YNBH11]:

$$R'(x) = \frac{\sum w_i(x)(R(u_i(x)) - \widehat{R})}{\sqrt{\sum w_i^2(x)}} + \widehat{R} \quad (7)$$

where $R'(x)$ is the pixel to compute, \widehat{R} is the mean of the texture sample, and $u_i(x)$ is the texture mapping.

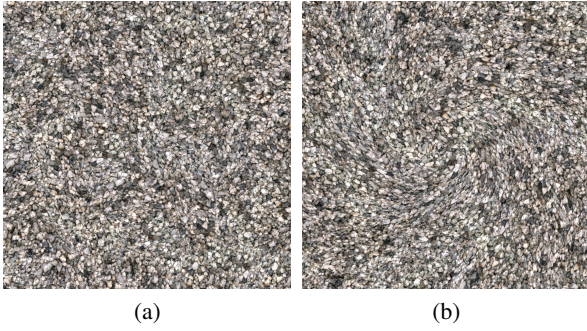


Figure 8: 2D rotational flow using a gravel texture exemplar, at frame 1 (a) and 240 (b).

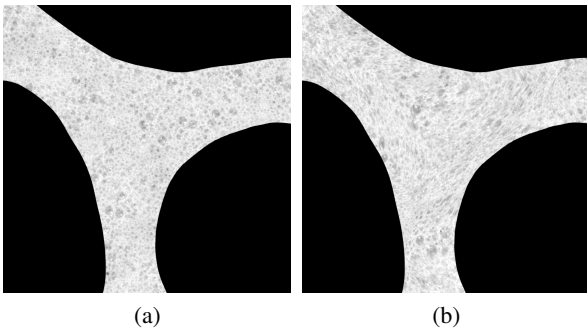


Figure 9: 2D split using a bubble texture exemplar, at frame 1 (a) and 240 (b).

All the pixels are computed on a texture atlas. We use the approach of Lévy [LPRM02] to generate the texture mapping between the atlas and the fluid surface. The results of the blending are illustrated in Fig. 8.

7. Results

We tested our approach using several fluid simulation scenarios similar to those from related work:

- 2D rotation flow, as in the paper of Yu et al. [YNBH11] (Fig. 8)
- 2D with split and merge (Fig. 10)
- 2D flow with split, as in the paper of Yu et al. [YNBH11] (Fig. 9)
- 3D viscous drop, as in the paper of Gagnon et al. [GDP16] (Fig. 13)
- 3D liquid dam break, as in the papers of Kwatra et al. [KAK*07] and Gagnon et al. [GDP16] (Fig. 14)
- 3D lava drop, as in the paper of Gagnon et al. [GDP16] (Fig. 12c)

Table 2 summarizes the statistics for the scenarios shown in the paper and video.

We compared our results with the method of Kwatra et al. [KAK*07]. Their method works best with small exemplars (for example, resolutions 64×64 or 128×128). Using larger exemplars has a severe negative impact on computation times, and complicates the selection of the window size for the best match search,

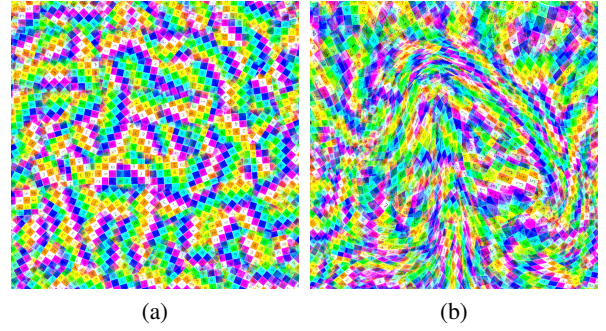


Figure 10: 2D fluid with split and merge, at frame 1 (a) and 140 (b).

Table 2: Statistics of our examples. “Nb poly” is the average number of polygons on the fluid’s mesh, “Nb patch” is the average number of patches, and “Atlas res” is the resolution (in pixels) of the computed atlas. Computation times for the Poisson disk creation, patch creation, and atlas creation are in seconds. The patch advection is negligible, taking less than 0.6 seconds for all of our examples. The patch and atlas creation steps are computed in parallel (on 20 cores in our tests). Our computations were conducted on a 2.80 GHz Intel Xeon® E5-2680 CPU.

	Nb poly	Nb patch	Atlas res	Poisson disk	Patch	Atlas	Total
Dam Break	7k	6900	$6k^2$	15	30	45	90
Viscous drop	28k	1740	$6k^2$	10	4	101	115
Split & merge	5k	674	$1k^2$	3	1.1	2	6.1
Split Y	6k	584	$1k^2$	0.5	1	1.3	2.8
Lava	51k	5700	$6k^2$	40	4	142	186

since the size of the window is related to the size of the patterns to replicate. Larger window sizes, required for larger exemplars, in turn also have a negative impact on computation times. For our method, either small or large exemplars can be used with negligible impact on computation time. As noted by Jamriska et al. [JFA*15], the method of Kwatra et al. [KAK*07] does not work for all texture exemplars, as it can lead to a wash-out effect, where the synthesized texture becomes too homogeneous and blurry. As can be seen in the video, the method of Kwatra et al. works better than ours when dealing with structured patterns. On the other hand, stochastic and isotropic textures are one of the causes of the wash-out observed in the synthesized results of Kwatra et al.’s method. In the accompanying video, we can see that our method preserves more of the exemplar colors and structure, as compared to the washed-out result from Kwatra et al.’s method.

We also compared our results with those of Gagnon et al. [GDP16] (Fig. 12). In their results, we can sometimes identify individual patches. With the large and rigid patches they use, we can observe blocky artifacts, and the optical flow is less respected. In comparison, with our approach, it is difficult to guess the boundaries of the patches, and very fine texture details remain visible in our output. In addition, we maintain a correspondence between

the texture movement and the free surface movement despite using large patches. As can be seen in the figures, as well as the

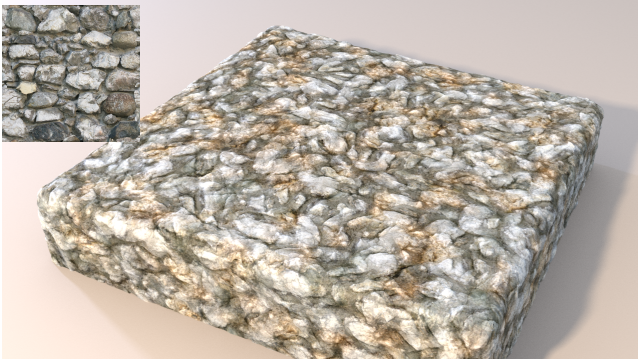


Figure 11: Dam break with stone texture exemplar (inset).

accompanying video, our approach works well on 2D examples (Fig. 8); with 3D viscous fluids (Figs. 12c and 13); and with 3D liquids (Fig. 14). We can also synthesize different textures using the same deformable patch distribution, simply by changing the texture exemplar (Fig. 13).

We should take particular note of Fig. 14, where the simulation has produced a complex mesh with a fine structure. The texture covers the mesh seamlessly and with good fidelity to the exemplar. In the accompanying video, it can be seen that our approach copes well with the entire animation despite the presence of multiple topological changes in the splashing liquid.

With the deformable patch representation, the texture conforms well to the movement of the fluid surface irrespective of any topological changes. We further demonstrate this in Fig. 15 and in the accompanying video, where we compare the fluid's velocity field (shown in blue) with the optical flow of the rendering (shown in green). The two fields are in close agreement, thus showing that the texture synthesis is able to match the liquid flow.

Thanks to the patch advection and Poisson distribution update, there are no regions with excessive numbers of patches or severe distortion. Moreover, with the density-based fading, we have fewer patches, and thus faster computation and fewer ghosting artifacts.

An added benefit of our approach is that we can apply other texturing techniques on fluids. For example, Fig. 1, 2, 11, 12c, and 14 use displacement mapping, while Fig. 13 uses normal mapping. Corresponding scenarios found in the accompanying video also use displacement and normal mapping.

7.1. Limitations

One limitation of our approach appears in scenarios where the fluid stops or barely moves. Patches will continue to fade in and out for several frames, even if the velocity is very small or null. The fading sometimes leads to the impression that the fluid continues to move slightly. In the video of the rotational flow (Fig. 8), the slow

movement makes it easier to note the fading in and out near the center.

Our approach also exhibits lengthy computation times; each frame can take a few minutes to compute, depending on the complexity of the fluid's animation. The advection and the patch creation times are similar to those of Gagnon et al. [GDP16]. However, the texture synthesis time can be more significant. Indeed, since for each output texel we are using a blending of all overlapping patches, we need to compute the exact position of the texel on the 3D surface. With this 3D texel position, we perform a projection through all intersecting patch triangles to get the uv coordinate that will provide the patch's texel. The large number of required projections acts as a bottleneck for the overall texture synthesis process.

Even with the density-based fading, some scenarios still produce visible ghosting effects. Just as with Yu et al.'s method [YNBH11], this is inevitable because blending is used to combine the color of the patches overlapping the same region of the surface. The contrast-preserving blending (Eq. 7) reduces blurring at the expense of increased ghosting. Regular blending could be used when results with less ghosting are preferable. The ghosting is also worsened with structured textures, as illustrated in Fig. 16.

8. Conclusion

We have presented an approach for synthesizing texture on the free surface of animated fluids. The proposed approach creates textured patches with limited distortion, based on a Poisson disk distribution. We want a good distribution for the specific case of fluids with small details, such as splashes and thin films, and we need to account for patches that will be wrapped on the fluid's surface. The update of the distribution and of the patches fully supports topological changes. We presented a more elaborate temporal fading out of patches, which allows control over their accumulation; this is most noticeable in areas where the flow converges or where different regions of the surface come into contact. Finally, we showed that the surface texture follows the velocity field of the fluid simulation, demonstrating this by comparing the optical flow with the fluid velocities.

We believe this work can be extended to include a measure of local velocity in our adaptive fading, further improving the texture evolution. The blending function is another area of potential development, further reducing ghosting beyond the improvements seen in this paper. Another avenue to reduce the ghosting could involve an extra step using a best-match search as in the method of Kwatra et al. [KAK*07], applied only in regions with excessive ghosting.

9. Acknowledgements

This work was funded by Digital District Canada, MITACS, Prompt, and ÉTS. We gratefully acknowledge the involvement of the former Digital District R&D team. We thank SideFX for providing Houdini licenses, and Hippolyte Mounier, from Photosculpt, for sharing some of the texture exemplars used in this project. The collaboration between ÉTS and Carleton University resulted from the 2017 Bellairs Multidisciplinary Computer Science Workshop held at the Bellairs Research Institute of McGill University.

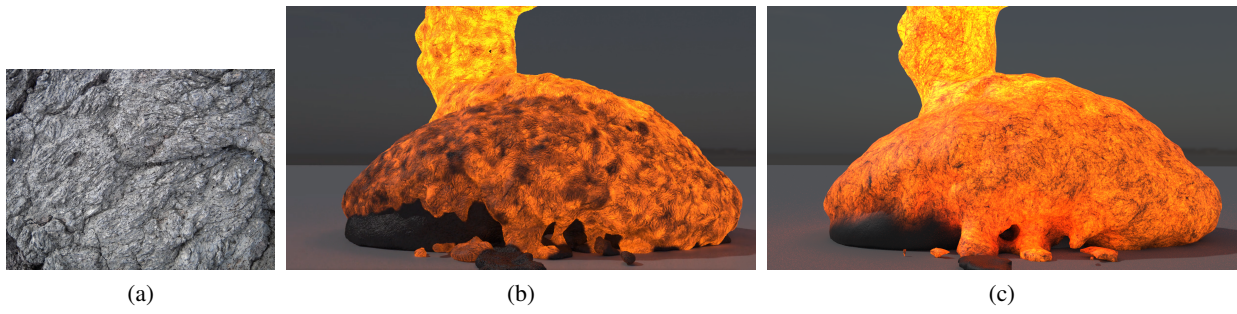


Figure 12: Lava simulation. Using texture exemplar (a), we compare between (b) Gagnon et al. [GDP16] and (c) our approach.

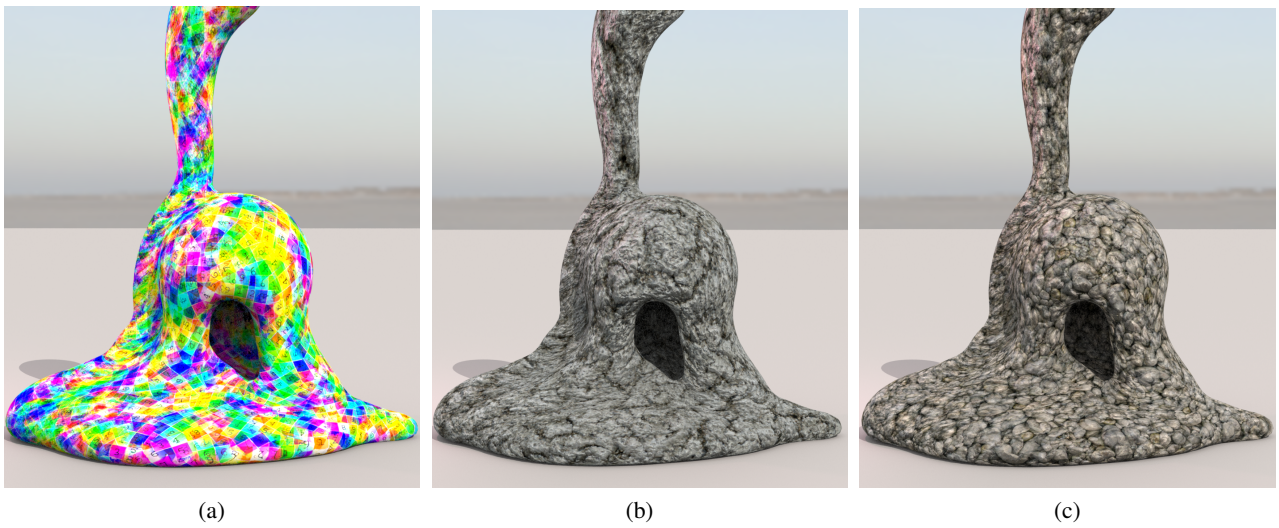


Figure 13: Different textures using the same underlying patch distribution.

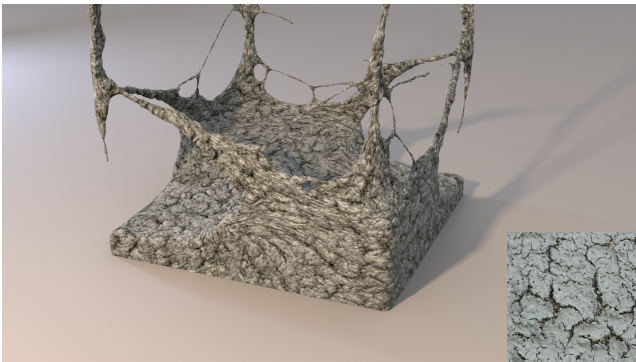


Figure 14: Dam break example with splash filaments.

Appendix A: Input variables

Tables 3 and 4 describe the parameters of our approach.

Table 3: Input variables

Variable	Description
d	Poisson disk radius
pa	Poisson disk plane angle
pva	Patch vertex angle threshold
δ_{max}	Is the maximum deformation allowed, we use 3
s	Scaling of the uv coordinates
mpt	Maximum projection threshold to handle topological changes
τ	Fading speed
cs	Ellipsoid thickness representing the smallest detail

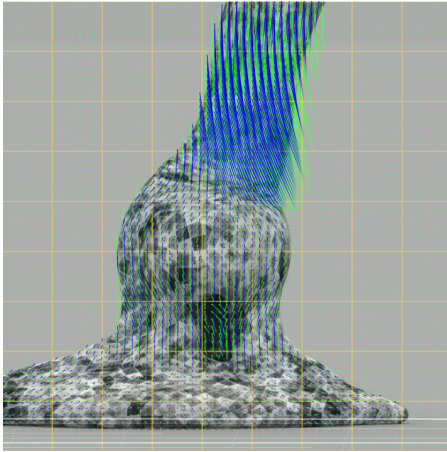


Figure 15: Test case demonstrating that the optical flow extracted from the texture advection matches the velocity field from the fluid simulation.

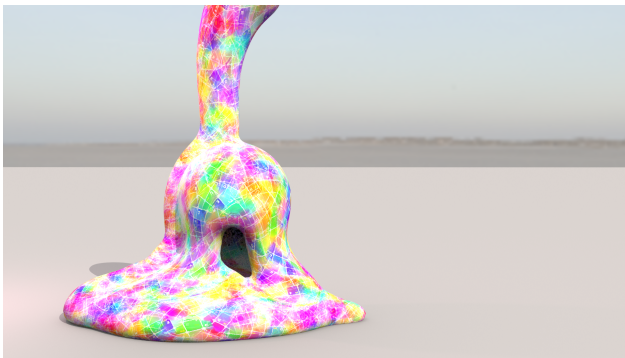


Figure 16: Ghosting artifacts can become visible, especially when using a structured texture.

Table 4: Constants

Constant	Description
$(1 - \alpha)d$	Kill distance where α is equal to 0.25
Q_{vmin}	Quality vertex min is equal to 0.5

- [LPRM02] LĂLVY B., PETITJEAN S., RAY N., MAILLOT J.: Least squares conformal maps for automatic texture atlas generation. In *IN SIGGRAPH 02 CONFERENCE PROCEEDINGS* (2002), pp. 362–371. [3](#), [5](#), [7](#)
- [NKL*07] NARAIN R., KWATRA V., LEE H.-P., KIM T., CARLSON M., LIN M.: Feature-guided dynamic texture synthesis on continuous flows. In *EGSR '07* (2007). [2](#), [3](#)
- [SAC*99] STORA D., AGLIATI P.-O., CANI M.-P., NEYRET F., GAS-CUEL J.-D.: Animating lava flows. In *Proceedings of the 1999 Conference on Graphics Interface '99* (1999), Morgan Kaufmann Publishers Inc., pp. 203–210. [2](#)
- [SCOGL02] SORKINE O., COHEN-OR D., GOLDENTHAL R., LISCHINSKI D.: Bounded-distortion piecewise mesh parameterization. In *Proceedings of the conference on Visualization '02* (Washington, DC, USA, 2002), VIS '02, IEEE Computer Society, pp. 355–362. [5](#)
- [WLKT09] WEI L.-Y., LEFEBVRE S., KWATRA V., TURK G.: State of the art in example-based texture synthesis. In *Eurographics 2009, State of the Art Report, EG-STAR* (2009), Eurographics Association. [2](#)
- [YNBH11] YU Q., NEYRET F., BRUNETON E., HOLZSCHUCH N.: Lagrangian texture advection: Preserving both spectrum and velocity field. *IEEE Trans. Visualization and Computer Graphics* *17*, 11 (Nov. 2011), 1612–1623. [2](#), [3](#), [5](#), [6](#), [7](#), [8](#)

References

- [BSM*06] BARGTEIL A. W., SIN F., MICHAELS J. E., GOKTEKIN T. G., O'BRIEN J. F.: A texture synthesis method for liquid animations. In *ACM SIGGRAPH 2006 Sketches* (2006), SIGGRAPH '06, ACM. [2](#), [3](#)
- [BZ17] BARNES C., ZHANG F.-L.: A survey of the state-of-the-art in patch-based synthesis. *Computational Visual Media* *3*, 1 (Mar. 2017), 3–20. [2](#)
- [DGP17] DAGENAIS F., GAGNON J., PAQUETTE E.: Detail-preserving explicit mesh projection and topology matching for particle-based fluids. *Computer Graphics Forum* *36*, 8 (2017), 444–457. [5](#)
- [GDP16] GAGNON J., DAGENAIS F., PAQUETTE E.: Dynamic lapped texture for fluid simulations. *Vis. Comput.* *32*, 6-8 (June 2016), 901–909. [2](#), [3](#), [7](#), [8](#), [9](#)
- [JFA*15] JAMRIŠKA O., FIŠER J., ASEANTE P., LU J., SHECHTMAN E., ŠYKORA D.: Lazyfluids: Appearance transfer for fluid animations. *ACM Trans. on Graph.* *34*, 4 (2015). [2](#), [7](#)
- [KAK*07] KWATRA V., ADALSTEINSSON D., KIM T., KWATRA N., CARLSON M., LIN M.: Texturing fluids. *IEEE Trans. Visualization and Computer Graphics* *13*, 5 (2007), 939–952. [2](#), [3](#), [7](#), [8](#)
- [KEBK05] KWATRA V., ESSA I., BOBICK A., KWATRA N.: Texture optimization for example-based synthesis. *ACM Trans. Graph.* *24* (July 2005), 795–802. [2](#)