Published: 10 December 2021

RESEARCH-ARTICLE

Generalized fluid carving with fast lattice-guided seam computation

SEAN FLYNN, Brigham Young University, Provo, UT, United States

DAVID HART, Brigham Young University, Provo, UT, United States

BRYAN STUART MORSE, Brigham Young University, Provo, UT, United States

SETH HOLLADAY, Brigham Young University, Provo, UT, United States

PARRIS KARL EGBERT, Brigham Young University, Provo, UT, United States

Generalized Fluid Carving With Fast Lattice-Guided Seam Computation

SEAN FLYNN, DAVID HART, BRYAN MORSE, SETH HOLLADAY, and PARRIS EGBERT,
Brigham Young University, USA

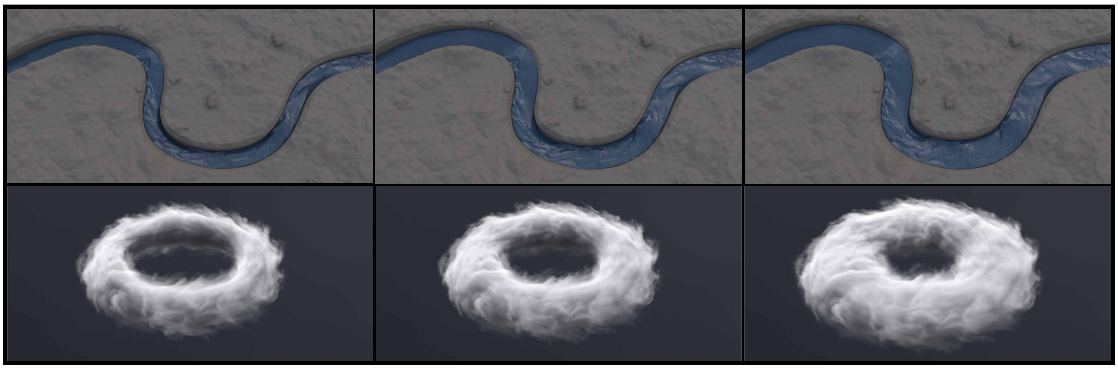

Fig. 1. We present a novel method for intelligently resizing a variety of volumetric data including this meandering river (top row) and smoke vortex (bottom row). In contrast to previous methods, our approach supports retargeting volumes with non-rectangular boundaries and motion. These simulations (middle column) are shown with a reduced size (left column) and an increased size (right column).

In this paper, we introduce a novel method for intelligently resizing a wide range of volumetric data including fluids. Fluid carving, the technique we build upon, only supported particle-based liquid data, and because it was based on image-based techniques, it was constrained to rectangular boundaries. We address these limitations to allow a much more versatile method for volumetric post-processing. By enclosing a region of interest in our lattice structure, users can retarget regions of a volume with non-rectangular boundaries and non-axis-aligned motion. Our approach generalizes to images, videos, liquids, meshes, and even previously unexplored domains such as fire and smoke. We also present a seam computation method that is significantly faster than the previous approach while maintaining the same level of quality, thus making our method more viable for production settings where post-processing workflows are vital.

CCS Concepts: • **Computing methodologies** → **Simulation tools**; **Physical simulation**; *Computer vision.*

Additional Key Words and Phrases: Volume retargeting, simulation control, post-processing, seam carving, content-aware scaling, fluid simulation

ACM Reference Format:
Sean Flynn, David Hart, Bryan Morse, Seth Holladay, and Parris Egbert. 2021. Generalized Fluid Carving With Fast Lattice-Guided Seam Computation. *ACM Trans. Graph.* 40, 6, Article 255 (December 2021), 15 pages. https://doi.org/10.1145/3478513.3480544

Authors' address: Sean Flynn, ecthelisean@gmail.com; David Hart, davidhart100@gmail.com; Bryan Morse, morse@cs.byu.edu; Seth Holladay, seth_holladay@byu.edu; Parris Egbert, egbert@cs.byu.edu,
Brigham Young University, Provo, UT, 84602, USA.

0730-0301/2021/12-ART255 $15.00
https://doi.org/10.1145/3478513.3480544

1 INTRODUCTION

Visual effects enable story tellers, artists, designers, and content creators to produce compelling entertainment that would not be possible otherwise. Recent research and development efforts have brought the ability to create these effects to the masses, both professional and amateur. However, post-production processes for controlling and editing large animated 4D volumetric data, like smoke, fire, and water, remain limited. Because this aspect of visual effects production is difficult and costly, research in this area remains highly influential and relevant.

Fluid carving [Flynn et al. 2019] was presented as a method for intelligently resizing particle-based liquid simulation data. This method was based on seam carving [Avidan and Shamir 2007] and used a graph-cut technique [Rubinstein et al. 2008] to compute and add or remove volumetric seams based on an energy function. Fluid carving provided a new way for users to control, reuse, and modify fluid simulations as a post-process without needing access to the simulation setup or initial conditions. However, that system was quite limited. For example, it only supported particle-based liquids, it only worked well for fluids with rectangular, axis-aligned boundaries and motion, and the graph-cut-based seam computation method was too slow for production use.

Our method overcomes these limitations. Rather than allowing liquids only, we provide support for a much wider range of volumetric data. We introduce a non-uniform lattice structure to guide the fluid carving process, thus allowing artists to retarget the dynamic shapes and motion that are characteristic of fluids. Using a greedy approach, we compute seams significantly faster than the previous method without reducing visual quality. These improvements make our system far more impactful in production scenarios, giving artists an unprecedented level of post-processing control over a wide range of large, complex FX data.

Our technique uses sequences of voxel grids represented as VDBs, an industry-standard volume format [Museth 2013], to provide a more general approach that can retarget any data that can be represented as a VDB. We demonstrate our method on images, particle-based liquids, smoke, fire, and even polygonal meshes. We leverage the energy functions from previous methods and also provide new energy functions for volumetric simulations like smoke and fire. Because we represent energy as VDBs, users have the flexibility to craft and manipulate them for their specific scenarios using the versatility of the OpenVDB API.

Seam carving was originally intended for retargeting images from one rectangular aspect ratio to another. Because there is no one-size-fits-all image resolution for the diversity of modern display formats, seam carving provided a useful new way to retarget images without losing important visual data. However, unlike images, fluids and other volumetric FX data rarely conform to rectangular boundaries. The diversity of boundary conditions for fluids is therefore much larger than that of images, and a more flexible, directable method that covers a wider range of shapes and motion is needed. Imagine a meandering river with its curved boundaries, or a flame that dynamically flickers and changes over time (Fig. 1). Resizing scenarios like these is ineffective if not impossible using the previous fluid carving approach because it assumes rectangular boundaries.

We introduce the use of non-uniform lattices to guide seam computations along non-rectangular boundaries. A lattice provides a mapping between a region of interest in world space and a uniform, rectangular space we call *lattice index space*. Fluid carving is performed in the usual manner on the lattice index space and then the region of interest can be remapped to update its shape and motion. Our method gives users a level of control that is not possible with previous methods, allowing them to resize volumes with a much wider range of motion, shapes, and boundaries.

To retarget isolated volume regions without resizing the global boundaries, we present a method for carvings seams in pairs. Normally when a seam is removed, the voxels along the seam are removed, and then all of the voxels on the positive side of the seam are shifted back toward the seam. This reduces the total size of the volume in the respective dimension by one voxel. With seam pairs, we remove one seam and add the other seam to account for the volume lost during the removal, or vice-versa. This process effectively retargets the region between the two seams without modifying the global volume boundaries.

A fundamental limitation of the original fluid carving system is that the graph-cut-based seam computation method is too slow for production use. While the work of Flynn et al. made computing seams on large 4D volumes feasible, seam computation was still the bottleneck, with computation times as high as 900 seconds per seam with all of the optimizations enabled. We present a greedy seam computation method that is orders of magnitude faster than the previous method. In our testing, the runtime of our greedy approach was up to 500 times faster. Additionally, we conducted a quantitative analysis and a user study that showed the same level of visual quality was maintained. With this approach, we can retarget volumes with a higher resolution in significantly less time than their approach. This opens up a wide range of new FX editing capabilities that were not possible with previous approaches.

1.1 Contribution Summary

In summary we make the following contributions:

- We introduce a novel fluid simulation retargeting method that allows artists to edit simulations containing a variety of different data including images, smoke, fire, particles, liquids, and meshes.
- We present energy functions for volumetric simulation data like smoke and fire.
- We introduce the use of lattice-guided seam computation to give users the control to more effectively resize volumes with non-axis-aligned motion and non-rectangular boundaries.
- We provide a method for adaptive resizing of isolated volume regions while maintaining global boundaries.
- We greatly reduce the runtime of seam computations on 4D volumes with a greedy approach while maintaining a similar level of quality. We verify this quality with both a quantitative analysis and a user study.

Our technique is more versatile, supports higher fidelity data, and is significantly faster than the approach given in [Flynn et al. 2019]. The result is a tool that will allow VFX artists and content creators to more effectively post-process and reuse FX data which can cut costs and enable new and more efficient workflows.

2 RELATED WORK

2.1 Seam Carving

In 2007, seam carving was introduced as a method for image retargeting [Avidan and Shamir 2007]. For an image, a seam is a sequence of pixels that runs from one side of the image to the other through consecutive, neighboring pixels. Since its introduction, many additional works have improved on the original image seam carving method and we direct the interested reader to a comprehensive survey of this line of work [Senturk and Akgun 2019].

Image seam carving was quickly extended to be able to retarget videos [Rubinstein et al. 2008]. A video can be treated as a 3D volume, where each frame is a 2D slice of the volume. A low-energy surface found in the volume using a method such as graph cut becomes the seam that is added or removed. Additional improvements and applications of video carving include video summarizations [Chen and Sen 2008], a multi-operator approach [Rubinstein et al. 2009], shape preserving methods [Wang et al. 2014], and the use of discontinuous seams [Grundmann et al. 2010].

In 2019, seam carving was again extended to 4D fluid simulation data [Flynn et al. 2019]. Similar to the original video carving work, Flynn et al. used graph cut to find the optimal seam through the 4D data. Like them, we use Maxflow version 3.04 [Boykov and Kolmogorov 2004] when computing graph-based seams.

Other retargeting approaches have been explored for images, such as patch-based methods [Barnes et al. 2009; Simakov et al. 2008]. Recently, deep learning approaches have been used to retarget images in a self-supervised manner [Cho et al. 2017; Tan et al. 2019], by using reinforcement learning on multi-operator search spaces [Kajiura et al. 2020; Zhou et al. 2021], or even by overtraining on a single image [Shaham et al. 2019; Shocher et al. 2019]. Although these methods have been used in the image domain, they do not easily extend to 3D and 4D data as training times increase and the

input data size becomes large and difficult to fit on the GPU. In comparison, the graph-cut method of seam carving has already been shown to effectively retarget higher dimensional data, so we focus on this approach.

Many improvements have been made on the speed and efficiency of seam carving methods. In the image domain, a wavelet tree can be used for computational efficiency [Hsin et al. 2014] or enhanced with GPU computation [Kim et al. 2015]. For 3D data, a multiple-seam graph cut approach has been used to decrease computation time [Dongfeng Han et al. 2009]. A GPU method has also been constructed for 3D seam carving [Chiang et al. 2009]. Additionally, a multi-pass dynamic programming approach has been proposed that is faster than the original graph cut method [Furuta et al. 2018]. Many of these works do not extend trivially to 4D representations. Additionally, large 4D simulation data makes it difficult to extend GPU methods, especially if a dense graph is required. We will show that our method is straightforward to extend to 4D and allows speed-ups similar to previous methods without requiring a high-end GPU.

A structure-aware seam carving method has also been explored [Yoon et al. 2014]. Similar to our method, this approach remaps the energy function to acquire gradients along some flow field. This method improved the preservation of the overall structure of image regions with well-defined edges.

To the best of our knowledge, this paper presents the first seam carving technique that allows the user to resize a region within an image by creating a lattice and using pairs of seams. This approach gives greater flexibility and control to seam carving-based retargeting when compared to the related work.

2.2 Guided Shape Editing

The use of non-uniform lattices to guide shape manipulations began with free-form deformation [Sederberg and Parry 1986]. This work has had a large impact on modeling and animation workflows and remains a common approach in production.

This line of work has continued with an extended sculpturing-tool approach [Coquillart 1990], an approach that used a Laplacian encoding of the surface [Sorkine et al. 2004], a technique that implicitly edits meshes based on the Poisson equation [Yu et al. 2004], and an approach that allows users to sketch curves to define a 3D model in a free-form manner [Nealen et al. 2007].

Our approach allows users to define curves and shapes that generate lattices which are then used to manipulate volumetric data. In contrast to the previous methods, we focus on content-aware data augmentation or reduction. Instead of shrinking or stretching the data, we remove unimportant data, or add data that maintains the characteristics of the unmodified data.

2.3 Volumetric Simulation

2.3.1 Fluid Simulation. The original fluid carving method [Flynn et al. 2019] is designed to work with liquid simulations that utilize the Fluid Implicit Particle (FLIP) method [Zhu and Bridson 2005]. FLIP itself builds upon previous Eulerian simulation methods [Stam 1999]. This seminal work was later extended to smoke [Fedkiw et al. 2001] and fire [Nguyen et al. 2002].

Like Flynn et al., we use the Houdini implementation of FLIP for liquids. We additionally use built-in Houdini methods for smoke and fire simulation. All these methods have been reviewed in detail [Bridson 2008]. Regardless of the data type, the result is converted to a VDB representation for use in our method [Museth 2013].

2.3.2 Simulation Efficiency Improvements. Many lines of research have focused on improving the computational efficiency of simulation methods. Grid representations such as octrees [Ando and Batty 2020; Losasso et al. 2004], tetrahedral meshes [Ando et al. 2013; Chentanez et al. 2007], SPGrid [Setaluri et al. 2014], and an adaptive staggered-tilted grid [Xiao et al. 2020] have been used as an alternative to uniform grids. Multi-grid methods [Briggs et al. 2000; McAdams et al. 2010], domain decompositions [Narain et al. 2008; Smith et al. 1996; Wicke et al. 2009], and GPU approaches [Harris 2005] have also been used.

Though these and other improvements continue to make simulation workflows more efficient, physical simulation continues to be computationally expensive and time consuming.

2.3.3 Fluid Post-Processing. Post-processing tools have been prevalent in the image and video domains for decades. In a similar fashion, some works have focused on editing previously made simulations. A set of level-set operators was presented to edit signed-distance representations of liquids and meshes [Museth et al. 2002]. Multiple liquid simulations can be blended together to make new simulations [Raveendran et al. 2014; Thuerey 2016]. Liquid simulations can also be integrated into existing fluid environments [Bojsen-Hansen and Wojtan 2016]. Smoke simulations can be modified by re-simulating a subspace of the fluid [Kim and Delaney 2013]. Sato et al. presented a technique for copying regions from one smoke simulation over to another smoke simulation [Sato et al. 2018a]. Deep learning approaches have also been used to synthesize new smoke simulations by training on existing simulations [Kim et al. 2019b] or even transferring style features between simulations [Kim et al. 2019a, 2020; Sato et al. 2018b]. A system for composing liquids using animation templates was also recently proposed [Schoentgen et al. 2020].

2.3.4 Directability and Control. In addition to editing previously made simulations, many works have focused on directing physical simulations towards a specific target to give more artistic control. Guide forces [Fattal and Lischinski 2004; Treuille et al. 2003], guide shapes [Nielsen and Bridson 2011], control particles [Foster and Fedkiw 2001; Rasmussen et al. 2004; Thürey et al. 2006], and optimal control theory [McNamara et al. 2004; Shi and Yu 2005] have been used. An interactive system has even been proposed to edit FLIP fluids during simulation keyframes [Pan et al. 2013]. A frequency-domain guiding approach that allowed high-resolution simulations to better match low-resolution scenarios with the same initial conditions was also recently presented [Forootaninia and Narain 2020].

As mentioned, we build upon the technique presented in [Flynn et al. 2019]. Our approach is more versatile because we support a larger range of volumetric data. We address the limitation of rectangular boundaries by providing an intuitive technique for guiding seams along non-uniform lattice structures. Finally, we significantly increase the production viability of retargeting fluids by computing seams significantly faster than the previous approach.

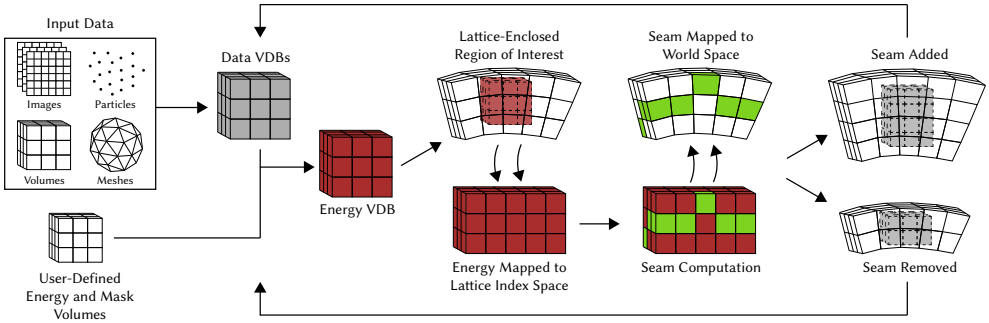

Fig. 2. An overview of our generalized, lattice-guided fluid carving method is shown. With our technique, users can retarget a variety of different data types. We provide the user the flexibility to provide custom energy and masks to manipulate the energy functions that are initially computed on the input data. The user can then enclose a region of interest in a non-uniform lattice structure to retarget volume regions without the restriction of rectangular boundaries.

3 LATTICE-GUIDED SEAM COMPUTATION

We begin with a brief review of the previous fluid carving method given by Flynn et al. to provide context for our method and to show why a lattice-guided approach increases the possibilities for retargeting fluids and other types of VDB data.

The fluid carving method of Flynn et al. accepts as input a set of particles defined at each frame of a simulation. A signed distance field (SDF) is computed using the particles on a 4-dimensional uniform rectangular grid. At each 4D voxel in the grid, an energy function is computed and stored. A 4D graph is then constructed where each node represents a voxel. The fourth dimension, which represents time, is important because it enforces temporal coherency when retargeting. Graph cut is then used to compute a seam that separates the volume into two sides. Particles are then added or removed and shifted, thus resizing the rectangular volume boundaries one voxel at a time.

While the approach of Flynn et al. provides a foundation for retargeting volumetric data, the nature of fluid data is inherently different from that of images, and simply extending image-based techniques to this more dynamic data is insufficient.

Seam carving requires seams that are monotonic. This monotonicity constraint ensures that seams have exactly one pixel in each row/column flowing from one edge of an image to the other along some axis. This guarantees that a seam carving operation modifies the width/height of an image by one pixel at a time. This constraint applies equally to volumetric data. However, what if a user would like to retarget a circular region of an image or volume using seam carving? The monotonicity constraint prevents seams that follow a circular path. If this constraint was simply removed to allow seams of arbitrary paths, it is not clear how to remove variable amounts of pixels or voxels from each row/column without large amounts of distortion. Additionally, there is no guarantee that shape silhouettes are preserved in such a process. This limitation would undermine the goals of seam carving.

In comparison, our novel lattice-guided seam carving method can effectively carve complex, non-rectangular volumetric data while preserving the benefits of the monotonicity constraint. Our method provides new and versatile ways to retarget image and volumetric data while avoiding the limitation of previous methods, as shown in Fig. 3. An overview of our method is shown in Fig. 2.

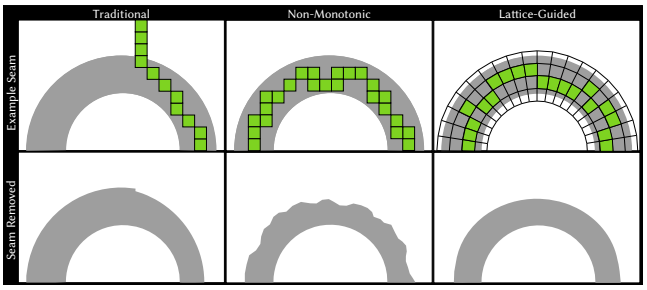

Fig. 3. The curved gray region with example seams (top row) is shown after removing one seam (bottom row). A traditional seam (left column) must flow monotonically from the bottom to the top, resulting in the region being resized on the right side only. Removing a seam computed without the monotonicity constraint (middle column) is not straightforward and results in silhouette distortion. Our lattice-guided approach (right column) allows seams that flow monotonically along the lattice, resulting in straightforward region retargeting with silhouette preservation.

3.1 Lattice Structure

We introduce the use of non-uniform lattices for fluid carving, which provides a mapping between a region of interest in world space and a uniform, rectangular space we call *lattice index space*. With this mapping, a user can use seam carving on non-rectangular regions, thus allowing a much wider range of retargeting applications.

We define our lattice structure as an ordered sequence of planes that each have an origin o_n, a set of unit-length basis vectors u_n, v_n, and w_n, and a set of vertices that are equally spaced one voxel length Δx apart. We call the sequence of plane origins the rig R of the lattice.

The distance between each plane origin o_n and the subsequent origin o_{n+1} is always Δx, i.e., $||o_{n+1} - o_n|| = \Delta x$. Each vertex of plane n is connected to its four neighboring vertices in plane n and also to the corresponding vertex in plane $n+1$, thus forming a lattice structure. The length of each lattice cell is guaranteed to be Δx in the u_n and v_n directions, but not in the w_n direction, except along the lattice rig. The length along w_n depends on the relative angle between planes n and $n+1$.

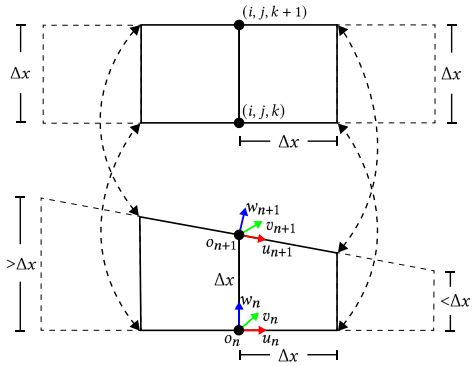

Fig. 4. A 2D slice of our lattice structure is shown with the mapping between lattice index space (top) and world space where the lattice is positioned (bottom).

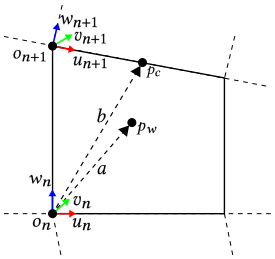

Fig. 5. A 2D slice of a world space point p_w is shown in relation to its enclosing lattice cell. Using the lattice origins, basis vectors, closest point p_c on plane $n+1$ to p_w, and the a and b vectors, p_w can be mapped to a lattice index space point p_l using Equation 3.

The lattice vertices store both a world space position and an (i, j, k) coordinate that provides a mapping to the rectangular lattice index space where cells are cubes with length Δx. The structure of our lattices is shown in Fig. 4.

3.2 Lattice Mapping Function

A lattice provides a mapping function f between a world space volume V and the rectangular lattice index space L:

$$f : V \rightarrow L \tag{1}$$

Similarly, we define g to be the inverse mapping function:

$$g : L \rightarrow V \tag{2}$$

A world space point p_w can be mapped forward to a lattice index space point p_l with $f(p_w) = p_l$ and back to world space with $g(p_l) = p_w$.

3.2.1 Forward Mapping. To map a world space point p_w that lies between lattice planes n and $n+1$ to a lattice index space point p_l, we first compute p_c, the closest point to p_w that lies on plane $n+1$. We then compute $a = p_w - o_n$ and $b = p_c - o_n$. We can then compute p_l as follows:

$$f(p_w) = p_l = (a \cdot u_n, a \cdot v_n, ((a \cdot w_n)/(b \cdot w_n) + n)\Delta x)) \tag{3}$$

A visual reference for this computation is shown in Fig. 5.

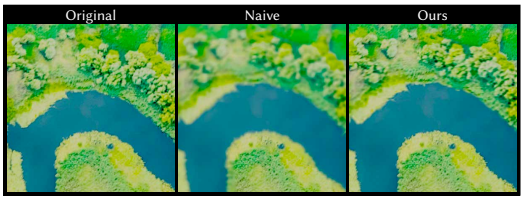

Fig. 6. An image of a river (left) is shown after increasing the size of the river by 10 lattice-guided seams using a naive mapping approach (middle) and using our mapping approach (right). Notice that the naive approach suffers from significant blurring artifacts.

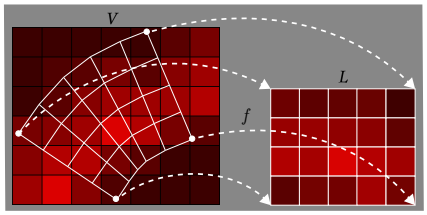

Fig. 7. A 2D slice of a lattice (shown in white) mapping energy (shown in red) from a world space volume V (left) to lattice index space L (right) where seams can be computed using traditional methods.

3.2.2 Inverse Mapping. A lattice index space point p_l can be mapped with g to a world space point p_w using the eight vertices of the enclosing lattice index space cell. These vertices store corresponding world space positions which can be trilinearly interpolated to determine p_w.

3.3 Carving the Mapping

With the mapping functions f and g described, we can now describe how we perform carving operations with our lattices.

The most straightforward but naive approach would be to simply map data values in V to L, compute and carve seams from these values in L, and then remap them back to V. However, due to the non-uniformity of the mapping, the values do not map perfectly between V and L, and this approach introduces significant blurring artifacts, especially in regions where the cells shapes differ widely between V and L. Performing multiple carving operations compounds these artifacts. Instead, we use a more sophisticated method where the values in V are updated using a carved mapping into a copy of V. This approach removes these issues, leaving only a slight blurring due to interpolation. The difference between the naive approach and our method is illustrated in Fig. 6.

We begin with the user defining a lattice around a world space region of interest. The processes for creating lattices will be described in Sec. 3.4. Using the lattice, the energy function which is defined on V is mapped to L with f as shown in Fig. 7.

A seam can then be computed with graph cut or our greedy approach (described in Sec. 4) using the energy function defined on L. Temporary copies L_c and V_c are created from L and V, respectively, and the seam is carved from L in the traditional manner. The carved lattice index space L^* provides a modified mapping function g^* which can be used to map each point $p_l = f(p_w)$ in L^* to a new point p_w^*:

$$p_w^* = g^*(f(p_w)) \tag{4}$$

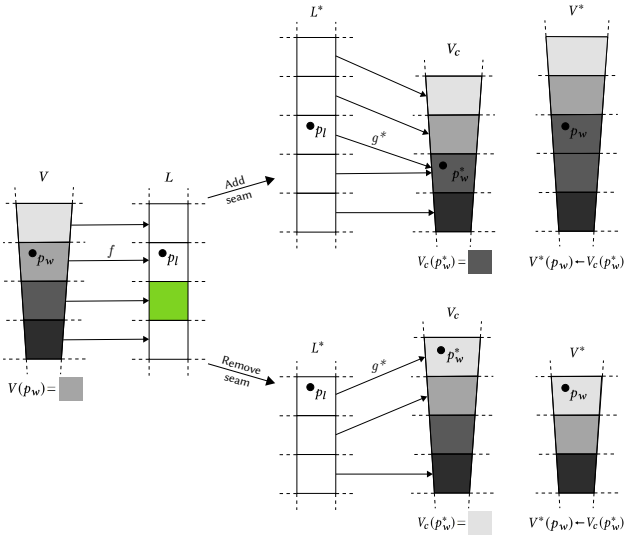

Fig. 8. The process for retargeting the world space volume V (left) to V^* (right) is shown in 2D for clarity. First, energy is mapped to lattice index space L (second from left) where a seam is computed and then carved yielding a modified mapping L^* (middle). The value at each point p_w in V can then be assigned to the value at the remapped point p_w^* in a copy of the volume V_c (second from right), thus yielding the retargeted volume V^*.

With the carved mapping, V is then retargeted to volume V^* by assigning the value at each point p_w in V to the value at each point p_w^* in V_c:

$$\forall p_w \in V : V^*(p_w) \leftarrow V_c(p_w^*) \tag{5}$$

With the carving operation complete, the original mapping L can be restored from L_c, V_c can be deleted, and V^* becomes V for the next carving operation. This process is shown in Fig. 8.

3.4 Lattice Creation

We provide two methods for creating lattices that can be used together to intuitively enclose and retarget a region of interest. As Flynn et al. noted, fluid carving works best when seams are carved parallel to the direction of greatest motion. Therefore, we provide a method for automatically computing the lattice so that it follows the direction of greatest motion. Alternatively, the user can define a static or animated sequence of points that form the lattice rig R, mentioned in Sec. 3.1. Using these two methods in conjunction, by first computing a rough lattice rig automatically, and then modifying it to the desired final configuration, provides a good balance between automation and user control.

3.4.1 Automatic Lattice Generation. When the simulation data contains a velocity field u, a lattice rig can be generated automatically that follows the motion of the simulation. We compute the lattice rig R by emitting and advecting a set of points starting from a user-defined location. At each frame, the points are advected, and a new set of points is added and advected. For each set of points, the centroid is computed, thus forming a sequence of points R that are centered about the simulation and follow its motion. R is then either modified as discussed in the next section, or is used to generate the final lattice structure. This method works best when the simulation

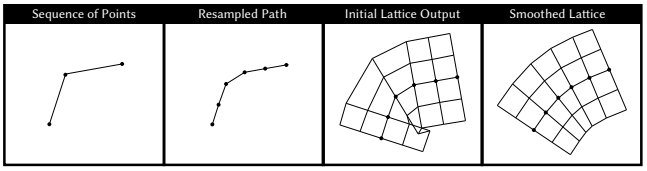

Fig. 9. A 2D slice of our method for lattice construction is shown. The user provides a path of manually or automatically defined points R (left), which is then resampled to voxel-length segments (second from left). A lattice is then generated using the resampled path (second from right). The lattice can then be smoothed to improve the mapping (right).

region of interest follows some overall path, curved or otherwise. The initial location for particle emission need not be static, as we support animated lattices that track with the simulation.

With the set of points R, we can generate the final lattice. First we resample the path formed by R, to ensure that all edges are of length Δx. Next we compute basis vectors such that for each point o_n, w_n points in the direction of the next point o_{n+1}, and u_n and v_n are orthogonal to w_n. The user may then optionally smooth the points by applying a mean filter in the u_n and v_n directions, thus aligning points with their subsequent points. The lattice vertices are then computed using the basis vectors and plane origins within a user-defined min and max boundary. An example of lattice construction is shown in Fig. 9.

The smoothing process is performed recursively until either there are no overlapping planes in the lattice, or a user-defined maximum iterations is reached. Though it is ultimately the responsibility of the user to avoid these issues, this process alleviates the burden on the user to avoid overlapping lattices. Lattices with overlapping regions will still work with our method, but may exhibit artifacts. The severity of these artifacts depends on the amount of overlapping region and on the visual importance of the simulation region where the overlap occurs. For example, in our rocket smoke trail example (Fig. 20), there are a few minor overlapping regions in the lattice, but the artifacts are negligible.

3.4.2 User-Defined Lattice Rigs. It is often clear to the user how the lattice should be defined. This is the case when the image, volume, or simulation region follows some predictable path, like a circle, or a curve. For example, a bagel in an image, or a flame that is confined to a round fire pit, clearly have circular shapes. In these cases, the user can either manually create a curve that follows the shape, or use a predefined shape like a circle or oval. The user can also animate these points based on the simulation scenario. This process is relatively straightforward using an application like Houdini. The set of points then forms R which can be used to generate the final lattice as previously described.

3.5 Cyclic Lattices

A common case for lattice-guided seam carving occurs when a region of simulation starts and ends in the same location, like our smoke vortex result shown in Fig. 16. Handling these cases requires special attention.

We provide an option to enable the construction of a cyclic lattice. This is done by connecting the last plane in the lattice to the

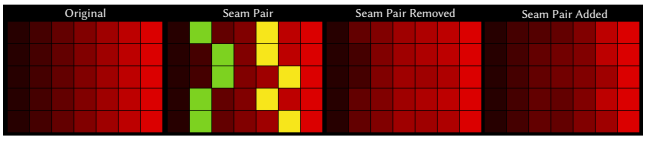

Fig. 10. A simple image of red values (left) is shown with a seam pair (second from left) being removed (second from right) and added (right).

first plane in the same manner that other lattice planes are connected. This ensures continuity between the regions at the end and beginning of the lattice when retargeting a volume.

3.5.1 Cyclic Graph-Cut Seams. To ensure that seams computed with graph cut are cyclic, we must connect the nodes at the end of the graph to the nodes at the beginning in the same way that neighboring rows/columns are connected to each other [Rubinstein et al. 2008]. Thus, when computing a cyclic graph cut, rather than separating the volume into two sides, e.g. the left side from the right, we are separating the outside from the inside of the cyclic region.

3.6 Seam Pairs

To resize a region of a volume without modifying the global boundaries, we present a method for computing and carving seams in pairs. When removing a seam, another seam is added to account for the region lost during the removal, and vice-versa. In this manner, the region between the two seams is retargeted while leaving the global boundaries unmodified.

Seam pairs are computed with two user-defined windows into the space being carved. One seam is computed with only the region inside the first window considered, and then a seam pair is computed in the second window. The windows must not be overlapping, and the pair window must be in the positive direction of the initial window. The first window is considered the region being retargeted, and the pair window will normally be positioned in an unimportant region. To decrease the size of the region of interest, the first seam will be removed in the traditional way, and the seam pair will be added, but in the opposite direction, with pixels being shifted in the negative direction into the region that was removed. To increase the size of the region of interest, this process is done in reverse. This is shown in Fig. 10. We also discuss and give examples of seam pairs in Sec. 6 in our meandering river and image results.

4 GREEDY SEAM COMPUTATION

In the original fluid carving work, seams were computed using a graph-based approach that guaranteed the optimal seam, but was too slow for production use, especially as simulations grew in resolution. Even when considering the graph in intervals, the computation was slow and sacrificed seam quality over the full frame range. In comparison, our method computes high quality seams over the entire frame range in a matter of seconds, making it a viable option for production workflows.

To increase the speed of the seam computation step, our method uses a greedy approach. When carving volumetric data, first, a 2D slice of the data is taken. Then, the lowest energy neighbor is selected in the next row. This process is repeated until the seam crosses the entire 2D slice. The next 2D slice is selected and this

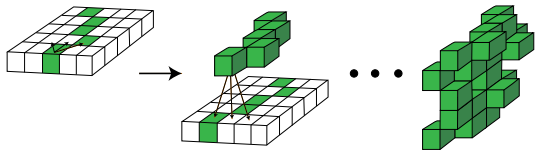

Fig. 11. When greedily carving 3D data, a seam is selected for the first 2D slice (left), then a seam is selected for the next 2D slice while enforcing adjacency to the previous 2D slice (middle). This process is repeated until the full seam is formed (right).

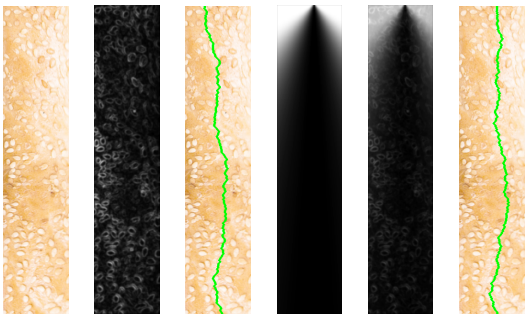

Fig. 12. An image in lattice index space (a) and its energy (b) result in a non-cyclic greedy seam (c). If an energy guide (d) is added to the energy (e), it results in a cyclic greedy seam that starts and ends in the same location (f).

process is repeated while enforcing that the seam be reachable by the previously selected seam. Repeating this process for all 2D slices makes a complete seam through the 3D data, as shown in Fig. 11. If the data is 4D, the entire previous process would be repeated for each 3D slice, while enforcing reachability between slices. To make our method more robust, multiple seams are generated in parallel by initializing them with different starting points along the first 2D slice. After all seams are generated, the one with the lowest energy through the volume is selected to be added or removed from the data.

4.0.1 Cyclic Greedy Seams. The greedy approach automatically works with the lattice-guided techniques described in Sec. 3. If the lattice is cyclic, however, the selected seam in each slice needs to be guided so that the final seam is reachable by the initial seam. This is done by adding an inverted multidimensional Gaussian to the energy function. The Gaussian is centered about the initial seam selection. The width of the Gaussian decreases as it steps towards the final seam. If the final width is small enough and the guide is scaled appropriately when added to the energy function, it guarantees the seam in the final 2D slice is reachable by the seam in the initial 2D slice, making the whole seam cyclic. A visualization of this process is shown in Fig. 12.

4.1 Quantitative Analysis

While the greedy-seams approach is much faster than the graph-based approach, it does not guarantee the optimal seam through the volume. We will show, however, that the calculated greedy seams

Fig. 13. When removing 100 seams from the original image (top left) the single-seam greedy approach will have some seams that have poor starting locations that lead it through the high energy area, leading to a poor retargeting (bottom left). In comparison, the multi-greedy approach, which computes multiple seams in parallel and selects the best one, removes this limitation, resulting in a retargeting (top right) that is comparable to the optimal seam carving approach (bottom right).

produce a visual result that is nearly identical to the optimal-seams result while being orders of magnitude faster to compute.

While the main focus of this work is volumetric data, doing a large-scale analysis of reasonably sized simulations would quickly become intractable in time. Thus, for this section, we focus our analysis on the image domain. We use 5000 images from the 2017 MS COCO validation dataset [Lin et al. 2014] for our quantitative analysis.

In the original seam carving work of Avidan et al., a dynamic approach was used to find the lowest energy seam in a given image. To guarantee an optimal single seam, it required analyzing the energy function over the entire image. Our greedy approach, in contrast, only searches a small space of the image and retrieves a local minimum in the energy function. In the original work, such an approach was avoided since globally high energy areas may inadvertently be selected based on the random starting position. However, we find this problem to be mostly avoided by computing multiple greedy seams in parallel and selecting the best one (which we refer to as the *multi-greedy* approach when comparing to the single-seam greedy approach). Suboptimal starting points are not used if a better seam is available. This simple but important step makes our seam carving approach visually comparable to the original dynamic method, as shown in Fig. 13.

To quantify these findings, we look at the overall energy in the image after each seam iteration. We report the average pixel energy in the resulting image. These measurements, averaged over our 5000 image dataset, are shown in Fig. 14. While the original method guarantees the optimal seam for any particular iteration and has the highest average pixel energy, the multi-greedy approach can lead to a different ordering of seams that still leads to comparable results, but at faster speeds.

From this analysis on image data, we can expect our greedy method to work for volumetric data with similar benefits. Additionally, we see even more pronounced computation-time differences between the greedy and optimal approaches as the greedy approach scales only linearly with dimensionality. We see speed-ups on the order of 80 to 500 times on our example simulations. These direct comparisons are given in Sec. 6.

Table 1. User Study Results. For each simulation (described in Sec. 6), participants were given videos of both the graph-cut method and our multi-greedy method in a side-by-side comparison. The order they were presented in was randomized for each of the 153 participants.

Scene Name	Graph-Cut	Multi-Greedy
Smoke Vortex	47%	53%
Meandering River	46%	54%
Bonfire	47%	53%
Wavy Torch	45%	55%

4.2 User Study

As a qualitative analysis, we conducted a user study with 153 participants via Amazon Mechanical Turk. Respondents were given pairs of images that compared different methods of seam carving, including our multi-greedy method. The images that were used came from from the same 2017 MS COCO validation dataset [Lin et al. 2014]. The subset that was used in the user study is given in the supplemental material.

As a baseline, we compared the original approach and our multi-greedy approach with the single-seam greedy method. As expected, both were selected more frequently than the single-seam greedy method, with the original method picked over single-seam greedy 72% of the time out of 248 samples and multi-greedy also picked over single-seam greedy 72% of the time out of 246 samples.

In the image domain, multi-greedy was picked over the original method 59% of the time out of 423 samples. Additionally, participants were given videos of simulations comparing the graph-cut method to our multi-greedy method and selected our method 54% of the time out of 918 samples. Per simulation statistics are shown in Table 1 (details about each simulation are given in Sec. 6). These numbers show that the multi-greedy method produces visual results that are just as good as the graph-cut method and may even be slightly preferable. This may be due to the fact that graph-cut can occasionally cut across the whole simulation to try and minimize the energy function, which creates very subtle artifacts, whereas our method generally avoids such cuts.

As noted in Sec. 2.1, recent retargeting methods can preserve semantic information in a way that the original energy-based seam carving method cannot. Our multi-greedy approach shares this limitation with seam carving, and so other more recent methods are still often preferable in the image domain when a lattice structure is not needed. Our method, however, has the flexibility to work on any domain and data where an energy function can be defined without requiring any other prior knowledge about the space it is operating on, making it preferable for the 3D and 4D data of interest in this paper.

5 GENERALIZED FLUID CARVING

The work of Flynn et al. only supported particle-based FLIP liquids which limited its versatility. Our new fluid carving system provides

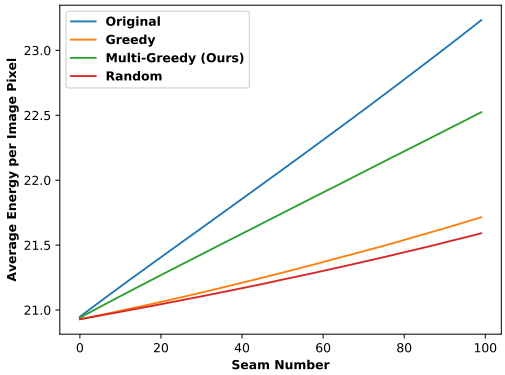

Fig. 14. Comparison of different seam carving methods averaged over 5000 images. For these images, removing 100 seams equates to removing approximately 20% of each image. Our method (Multi-Greedy) quickly computes 8 seams in parallel for each seam iteration and does comparably to the slower dynamic programming method with forward energy (Original), whereas the single-seam greedy approach (Greedy) does only slightly better than our baseline of removing random vertical seams from the image (Random).

support for VDB grids, extending the functionality of fluid carving beyond particles, to surfaces, volumetrics, and any other data that the user might want to represent as a VDB.

We will first describe how we handle each type of data along with the corresponding energy functions including new energy functions for volumetric simulations like smoke and fire. Then we describe how the user can craft and modify the energy functions to best suit their specific scenario. The result is a novel system that gives users a greater level of control over a wider range of applications in VFX production and in digital content creation.

5.1 Energy Functions

Seams are computed based on an energy function and will follow the path of least energy that separates the volume into two sides. Therefore, the quality of retargeting using a seam-carving approach is primarily dependent on the choice of energy function. For liquids, Flynn et al. noted that the surface is the most important visual aspect of liquids, thus mean curvature was used as the primary component of energy. Kinetic energy was also used to preserve high-velocity regions of the simulation. To support a wider range of data types, we must carefully choose energy functions that preserve the important visual information for each type of data.

Our generalized fluid carving approach supports any data that can be represented as a VDB. We specifically demonstrate results for images, particle-based liquids, polygonal meshes, smoke, and fire. We will now describe how we handle each type of data as a VDB and how we define energy on them.

5.1.1 Images. To support images, we first convert each image channel, e.g. RGBA, into a 2D VDB grid. For image energy E_i we use the gradient magnitude of the pixel intensities, the same energy function that was used in the initial seam carving formulation [Avidan and Shamir 2007]:

$$E_i(i,j) = \left\| \frac{\partial}{\partial x} I(i,j) \right\| + \left\| \frac{\partial}{\partial y} I(i,j) \right\| \tag{6}$$

where $I(i,j)$ is the intensity of the pixel at (i,j). This energy function could also be used for higher dimensional color data like videos or 3D textures. The gradient magnitude of intensities works well in most scenarios as documented in previous work.

5.1.2 Liquids and Meshes. For liquids, consisting of either particles or surfaces, and for meshes, we first compute a signed distance field (SDF) ϕ and store it as a VDB. Our surface energy function E_s consists of mean curvature and kinetic energy, the same as the previous fluid carving approach [Flynn et al. 2019]:

$$E_s(i,j,k,f) = |H(i,j,k,f)| + \lambda K(i,j,k,f) \tag{7}$$

where H is the mean curvature of the SDF ϕ

$$H = \nabla \cdot \nabla \phi \tag{8}$$

K is kinetic energy

$$K(i,j,k,f) = u(i,j,k,f)^2 \tag{9}$$

and u is the velocity field. f is the temporal coordinate, representing a frame of the simulation. We use λ to scale the relative weighting between H and K and to properly add quantities of different units. Note that we only need to worry about the relative weighting since E_s will undergo a minimization optimization. We let $\lambda = 1$ for our experiments.

To improve the quality of retargeting SDF data, we also allow the user to optionally apply a mean, median, or Gaussian filter with a user-defined size. When using a filter, the voxels directly along the seam are replaced with the filtered values. The neighboring values are not affected by the filter. This alleviates any discontinuities that can result from aggressively retargeting difficult scenarios.

5.1.3 Volumetrics. Volumetric simulations consist of a set of scalar fields S and a velocity field u. To preserve the defining curling, rotating nature of volumetric simulations like smoke and fire, we use the magnitude of the vorticity ω of the simulation when retargeting.

S represents various data like density, temperature, the presence of flame, etc. For a very simple smoke simulation, S might consist of a single scalar density field, but fire might include more fields. When rendering, the user assigns these fields to shader parameters, thus determining the final look of the rendered simulation. Therefore, the choice of energy function for volumetric simulation depends on how these fields are used. For our purposes, we assume that S is the set of scalar fields that affect the final look of the simulation as determined by the user. We take the gradient magnitude of S, similar to our image energy function E_i, to capture these features. The user can give relative importance to each field in S as will be described in 5.2. Our volumetric energy function E_v is then defined as follows:

$$E_v(i,j,k,f) = \|\omega(i,j,k,f)\| + \lambda_1 K(i,j,k,f) + \lambda_2 \sum_{s \in S} \|\nabla s(i,j,k,f)\| \tag{10}$$

where

$$\omega = \nabla \times u \tag{11}$$

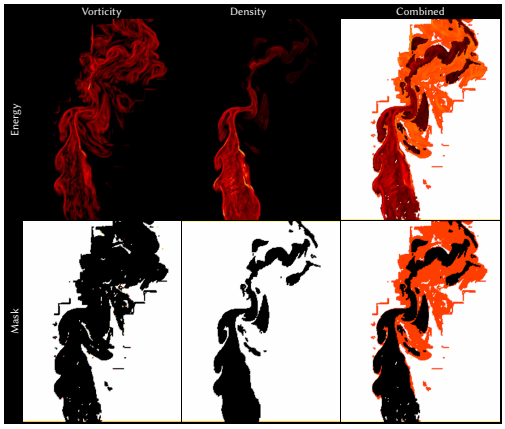

Fig. 15. An example of user control from our wavy torch result is shown. The energy functions (top) can be masked (bottom) to weight certain regions higher than others. In this case, the energy is weighted such that seams will focus first on the fire, second on the smoke, and last on the empty region.

We include kinetic energy in E_v as well to handle high-velocity volumetric simulations. Again, we let $\lambda_1 = 1$ and $\lambda_2 = 1$ for our experiments, but these could be adjusted as needed for various simulation setups.

With energy functions defined for each of these data types, the user has the flexibility to perform fluid carving on a variety of different data, making our approach applicable to most conceivable production scenarios.

5.2 User Control

Simulation data is crafted very carefully, often with very specific artistic direction. To make our system consistent with this approach, we must also allow a high degree of user control. Because we leverage OpenVDB [Museth 2013], a system that is familiar to most visual effects artists, the user can craft and manipulate energy functions for their specific data.

We illustrate this control with an example found in our wavy torch result (Sec. 6). Out of the box, using E_v would result in seams that simply avoid the flame and smoke entirely. Instead, using the built-in functionality of OpenVDB in Houdini, the user can easily create masks for the empty region, the smoke, and the fire itself. High energy can be assigned to the empty region, and the smoke and fire regions can be weighted so that seams will tend to focus first on the fire and second on the smoke. This way, seams will be carved primarily within the flame, as desired in this case. 2D slices of the energy function and its respective masks are shown in Fig. 15.

While most cases will work without needing this level of user control, rather than focus on automating the entire system, we provide the control that is expected in VFX production.

6 RESULTS AND DISCUSSION

We now present results on a number of different simulation scenarios including polygonal meshes, liquids, smoke, and fire. We show the parameter and timing information for our simulation results in Table 2. We also demonstrate results for images. In addition to the still images presented in this section, we direct the reader to view our results in motion in the included supplementary video.

All of our results except for the smoke plume were computed on a machine with an Intel Core i9-9900k CPU. The smoke plume results were computed with an Intel Xeon Gold 5118 which has a much lower per-core clock speed. Our method is implemented in C++ in the Houdini Development Kit (HDK) with TBB for multi-threading. We used the OpenVDB API [Museth 2013] heavily as our method acts on VDB grids. This includes SDF computation, filtering, volume compositing, and all aspects of our energy function creation. We utilize multi-threading for a few operations like seam removal/addition, lattice creation, and the remapping process. None of our methods use the GPU. We use the same graph-cut library [Boykov and Kolmogorov 2004] as the method we compare against [Flynn et al. 2019]. The simulations were authored using SideFX Houdini and the final results were rendered with Houdini's Mantra renderer. The images in Fig. 22 were obtained from and are used under license from Shutterstock.com. While we have taken a reasonable effort to optimize our method, there is still potential for additional runtime improvements with future work.

Many of our results include a comparison between our multi-greedy seam computation method and the previous graph-cut approach presented by Flynn et al. We utilize the optimizations presented by Flynn et al. in an effort to make the graph-cut runtimes as fast as possible while maintaining quality. We include the values we used in our results for the optimization parameters that had the most meaningful impact, k_i, and c, in Table 2. The keyframe interval k_i represents the number of frames that the simulation sequence is split into when computing the sequence of graph cuts. Nodes in the graphs are pruned below our energy threshold c. Before computing energy we also remap our energy functions to always be between 0 and 1. For multi-greedy seams computations, we set the number of greedy seams to compute equal to or less than the number of CPU threads, specifically we used 22 for the smoke plume and 16 for the rest of our simulation results. In practice, we do not need to set the c or k_i parameters as we found our multi-greedy method to be far more practical than graph cut. We provide the graph-cut results for comparison only. We also found that our system does not require significant trial-and-error parameter tuning, but we provide the option for user control as described in Sec. 5.2.

6.0.1 Smoke Vortex. We first show results for a smoke vortex simulation in Fig. 16. In this simulation, S consists of a single scalar density field. Our energy function E_v consists of vorticity energy ω and the gradient of the density field. These two components of energy were weighted equally and were not masked in any way. We omitted using kinetic energy K as it had no impact on the results. A circular, cyclic lattice, which was created from a Houdini circle curve primitive, was used to guide seams around the vortex shape. In this case, 18 seams were removed, and 30 seams were added.

This result illustrates that our method can handle a fast moving cyclic simulation. Because seam carving traditionally requires that seams flow monotonically along rectangular boundaries, computing seams in a circular shape like in this example would not be possible with the previous methods.

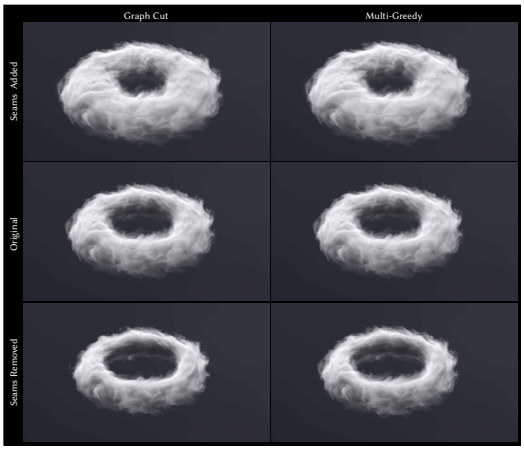

Fig. 16. Our smoke vortex results are shown with a comparison between the graph-cut and multi-greedy approaches. Our method allows retargeting simulations with a circular shape and cyclic motion. Notice that the faster multi-greedy approach maintains the same level of quality.

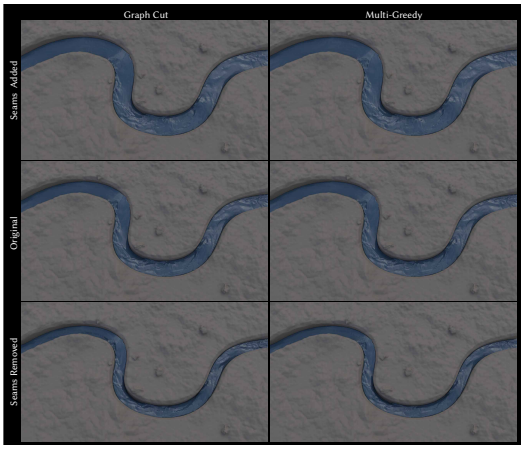

Fig. 17. Our meandering river results are shown with a comparison between the graph-cut and multi-greedy approaches. Notice that the riverbank remains tightly coupled to the river after retargeting.

6.0.2 Meandering River. Next we show results for a meandering river simulation in Fig. 17. This simulation consists of both a particle-based FLIP simulation for the river and a static polygonal mesh for the riverbank. We used seam pairs (Sec. 3.6) to carve both the river and riverbank simultaneously. Because the riverbank is static, a single seam is computed and paired with the animated seams computed in the water when carving. E_s was used for the energy function. A mask was used to remove energy below the surfaces of both the river and the riverbank. The lattice that enclosed the river and a portion of the riverbank was generated from a static curve drawn by the user. After each carve operation, the surface was filtered along the seam with a 3x3x3 Gaussian kernel. In this case, 13 seams were removed, and 13 seams were added.

In the case of a river with axis-aligned boundaries, like the one shown in the results section of [Flynn et al. 2019], adjusting the shoreline after retargeting the river is straightforward. The shoreline can simply be translated in one dimension to conform to the retargeted river. However, in this more complex meandering river example, the shoreline must conform to a new meandering shape. This example illustrates that our method eliminates the time-intensive manual process of updating non-rectangular boundaries to match a retargeted non-rectangular volume. By using seam pairs and a lattice, the region between the two seams is automatically updated and the boundary between the shore and river remains tightly coupled.

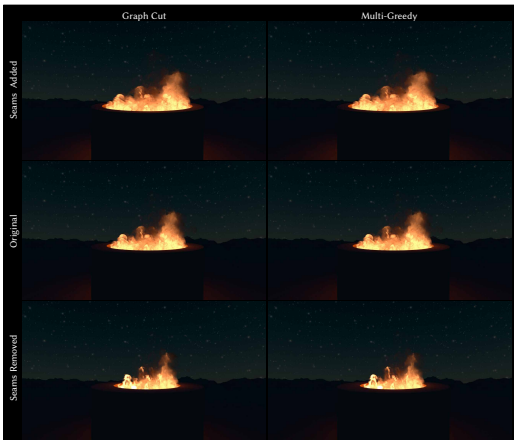

Fig. 18. Our bonfire results are shown with a comparison between the graph-cut and multi-greedy approaches. Similar to the smoke vortex, this example shows the result of carving cyclic seams, but this time with the round fire pit that resizes simultaneously with the simulation.

6.0.3 Bonfire. Our results for a bonfire simulation are shown in Fig. 18. This simulation features a round fire pit that was enclosed in a cyclic, circular lattice to carve seams in a circular fashion, thus resizing the whole boundary of the bonfire in all directions equally. This simulation has scalar fields for density, temperature, fuel, and flame, but we only used density and flame for S because they were the fields used in the shaders to determine the final rendered appearance. In this case, 16 seams were removed, and 10 seams were added.

6.0.4 Wavy Torch. Next we show results for a wavy torch simulation in Fig. 19. The energy function for this simulation was the same as that of the bonfire simulation. In this case, 10 seams were removed. As described in Sec. 5.2, to ensure that seams would not avoid the simulation altogether, we used masks to control the seam computations. The empty region was weighted by .8, the smoke .15, and the flame by .05. In this manner, the seams would tend to target the flame first, the smoke second, and the empty region last. The lattice was created using a slightly modified lattice rig R that was automatically computed using the velocity field as described in Sec. 3.4. This result demonstrates the level of control and flexibility that our generalized, lattice-guided approach gives the user.

6.0.5 Rocket Smoke Trail. To demonstrate that our lattice-guided method can retarget a more complicated simulation domain that curves in multiple directions, we show results on a smoke trail that follows the spiraling path of a rocket in Fig. 20. We used the same energy function for this simulation as the smoke vortex. The empty regions in this example were weighted with a mask similarly to the

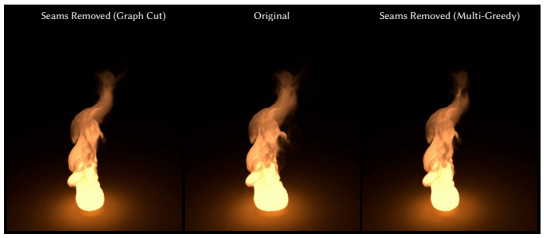

Fig. 19. Our wavy torch results are shown with a comparison between the graph-cut and multi-greedy approaches. This simulation uses an animated lattice to guide seams as the flame waves back and forth.

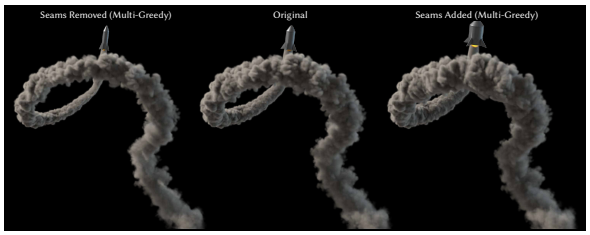

Fig. 20. This rocket smoke trail shows that our lattice-guided method supports simulation domains that follow paths that curve in multiple directions.

Fig. 21. This high-resolution smoke plume result shows that the multi-greedy approach can handle data that would be too large for graph-cut.

wavy torch example to ensure that seams did not avoid the smoke altogether. 10 seams were removed and added in this result.

6.0.6 Smoke Plume. Our final simulation result is shown in Fig. 21. The energy function for this result is identical to that of the smoke vortex and rocket smoke trail. We did not use a lattice in this case. Instead, this result is focused on testing the multi-greedy approach on a simulation that is too large for the graph-cut method. The keyframe interval optimization from Flynn et al. allows fluid carving to scale to simulations with long frame ranges, but in this case, computing a seam on even a single frame would takes hours or more. Therefore, carving this simulation would be intractable using the graph-cut approach. In contrast, our multi-greedy approach had no problem computing a seam quickly on the full 216 frames. In this case, 30 seams were removed, and 30 seams were added.

6.0.7 Images. We also demonstrate our method on the images shown in Fig. 22. We use the standard gradient magnitude energy function E_i. These results show the use of seam pairs (Sec. 3.6). One seam is carved in the region of interest, and another is carved in reverse just outside of this region, thus retargeting the region between the seam pairs rather than the entire image.

6.1 Limitations and Future Work

The prospect of fully adopting a fluid-carving-based approach into a production studio has potential for many interesting future works. Generalized fluid carving focuses on retargeting volumetric data. Geometric data like polygonal meshes or particles must first be rasterized to a volumetric SDF to use our technique. This works great for fluids or meshes that are assigned procedural materials, where converting back and forth between the SDF representation is straightforward. However, many scenarios involve meshes with UV-mapped textures. For example, in a real production scenario, the riverbed and riverbank in our meandering river example would have textures assigned that would need to be updated after applying fluid carving. Future work could explore simultaneously modifying the texture maps with a retargeting technique like seam carving to remove the need to manually update these textures. Furthermore, extending our technique to support directly retargeting meshes could be interesting.

We designed our lattice structure to be simple to create and intuitive to use for a wide range of interesting shapes. However, this simplified design limits our lattices to following a singular path that cannot overlap onto itself. This means that simulations that have multiple interacting regions with distinct, unaligned motions would be difficult with our method. For example, a river that branches into many tributaries, or a simulation with multiple interacting smoke plumes flowing in different directions would not be ideal for our method. Additionally, creating lattices that have abrupt changes in shape result in overlapping lattice regions where the mapping is not one-to-one. Future work could explore the use of more sophisticated lattice structures to better handle scenarios like these.

Because seam computation using our multi-greedy approach is so fast, there is potential for future work to improve the speed of other aspects of fluid retargeting to make it a truly interactive method. Like the previous method, our system remains a pre-process where the data for resizing is cached and then the user can interactively explore different retargeted sizes after the initial computation. However, because there is no longer a seam-computation bottleneck, GPU approaches could be applied to the remapping process, for example, to make it closer to real-time. Currently, the runtime of our method is dominated by the data I/O of loading and unloading simulation caches and letting the Houdini nodes bake.

Our post-carve remapping process (Sec. 3.3) introduces a small amount of blurring after each seam carving operation. This effect only becomes apparent after several carving operations. This is due to linearly interpolating values back and forth between world space and lattice index space. This can cause very small features to diminish or disappear after many successive carving operations. For example, in our smoke vortex results, the fine features along the silhouette are slightly softened after retargeting. This is also apparent when looking closely at the remapped regions in the image results. Higher order interpolation methods and lattice structures, or

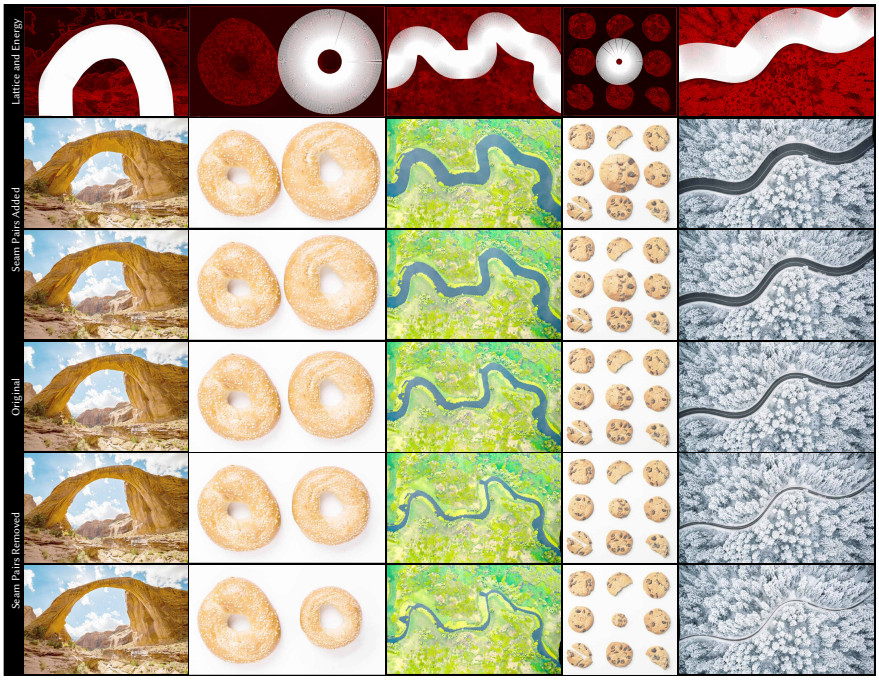

Fig. 22. Our images are shown with lattice-guided seam pairs added and removed. The top row shows the energy (red) and lattices (white) we used.

Table 2. Example Scene Results. The energy threshold c was used for both the graph-cut and multi-greedy approaches. Node Count denotes number of nodes of the generated graph when a frame interval of k_i was used for the graph-cut method. The average seam computation times for both the graph-cut and multi-greedy approaches are given in their respective columns. Lattice Remapping denotes the average time for mapping the world space volume V to the retargeted volume V^* after carving (Equation 5).

Scene Name	Dimensions	Frames	c	k_i	Lattice Remapping	Node Count	Graph Cut	Multi-Greedy
Smoke Vortex	322 x 105 x 331	84	0.085	3	462.6 sec	5353070.9	2076.4 sec	3.7 sec
Meandering River	903 x 79 x 390	159	0.0	3	509.8 sec	6232326.3	682.4 sec	5.2 sec
Bonfire	221 x 175 x 233	120	0.016	4	233.1 sec	4311588.6	752.8 sec	9.3 sec
Wavy Torch	222 x 360 x 95	72	0.0	3	82.9 sec	7806457.8	1453.7 sec	2.2 sec
Rocket Smoke Trail	317 x 493 x 409	72	0.0		137.6 sec			6.4 sec
Smoke Plume	300 x 595 x 220	216	0.001					71.9 sec

possibly post-carve sharpening filters could be explored to alleviate these issues.

As the number of image-based deep learning methods continues to increase, there is potential for more of these methods to be applied to volumetric data like fluids. Because most of these approaches are built around the assumption of rectangular boundaries like seam carving, our lattice-guided approach may provide potential for applying many of these techniques more effectively to non-rectangular shapes like fluids. For example, SinGAN [Shaham et al. 2019] and InGAN [Shocher et al. 2019] could be applied to a remapped fluid using our lattice-guided approach to provide a new retargeting method for fluids. Our lattice-guided approach could potentially open many new areas of research for volumetric data.

7 CONCLUSION

We have presented a novel method for intelligently resizing a wide variety of data including fluids. Our method is general in that it supports any data that can represented as a VDB grid. This allows content-aware resizing on almost any conceivable type of FX data including images, particles, liquids, smoke, and fire. Our lattice-guided approach enables retargeting on a much wider range of volumetric shapes and motions, thus removing the rectangular, axis-aligned restrictions of the previous seam-carving methods. Our multi-greedy approach to seam computation is orders of magnitude faster than the previously used graph-cut approach while resulting in retargeted data that we have shown to be quantitatively similar and that maintains the same level of visual quality.

Because producing compelling large volumetric data like fluids remains one of the most expensive and complicated parts of visual effects, animation, and video game production, post-processing techniques for this data are vital. With our method, artists and content-creators have a new intuitive and directable way to edit and post-process large volumetric data. This will reduce the need to resimulate and give more flexibility for directors to experiment late in the production process, thus reducing time and cost, and ultimately resulting in a better final product.

REFERENCES

Ryoichi Ando and Christopher Batty. 2020. A Practical Octree Liquid Simulator with Adaptive Surface Resolution. *ACM Trans. Graph.* 39, 4, Article 32 (July 2020), 17 pages. https://doi.org/10.1145/3386569.3392460

Ryoichi Ando, Nils Thürey, and Chris Wojtan. 2013. Highly Adaptive Liquid Simulations on Tetrahedral Meshes. *ACM Trans. Graph.* 32, 4, Article 103 (July 2013), 10 pages. https://doi.org/10.1145/2461912.2461982

Shai Avidan and Ariel Shamir. 2007. Seam Carving for Content-aware Image Resizing. *ACM Trans. Graph.* 26, 3, Article 10 (July 2007). https://doi.org/10.1145/1276377.1276390

Connelly Barnes, Eli Shechtman, Adam Finkelstein, and Dan B Goldman. 2009. PatchMatch: A Randomized Correspondence Algorithm for Structural Image Editing. *ACM Transactions on Graphics (Proc. SIGGRAPH)* 28, 3 (Aug. 2009).

Morten Bojsen-Hansen and Chris Wojtan. 2016. Generalized Non-reflecting Boundaries for Fluid Re-simulation. *ACM Trans. Graph.* 35, 4, Article 96 (July 2016), 7 pages. https://doi.org/10.1145/2897824.2925963

Y. Boykov and V. Kolmogorov. 2004. An experimental comparison of min-cut/max-flow algorithms for energy minimization in vision. *IEEE Transactions on Pattern Analysis and Machine Intelligence* 26, 9 (Sep. 2004), 1124–1137. https://doi.org/10.1109/TPAMI.2004.60

Robert Bridson. 2008. *Fluid Simulation.* A. K. Peters, Ltd., Natick, MA, USA.

William L. Briggs, Van Emden Henson, and Steve F. McCormick. 2000. *A Multigrid Tutorial: Second Edition.* Society for Industrial and Applied Mathematics, Philadelphia, PA, USA.

Billy Chen and Pradeep Sen. 2008. Video Carving. In *Eurographics 2008 - Short Papers*, Katerina Mania and Eric Reinhard (Eds.). The Eurographics Association. https://doi.org/10.2312/egs.20081022

Nuttapong Chentanez, Bryan E. Feldman, François Labelle, James F. O'Brien, and Jonathan R. Shewchuk. 2007. Liquid Simulation on Lattice-based Tetrahedral Meshes. In *Proceedings of the 2007 ACM SIGGRAPH/Eurographics Symposium on Computer Animation* (San Diego, California) *(SCA '07)*. Eurographics Association, Aire-la-Ville, Switzerland, Switzerland, 219–228. http://dl.acm.org/citation.cfm?id=1272690.1272720

Chen-Kuo Chiang, Shu-Fan Wang, Yi-Ling Chen, and Shang-Hong Lai. 2009. Fast JND-Based Video Carving with GPU Acceleration for Real-Time Video Retargeting. *IEEE Trans. Cir. and Sys. for Video Technol.* 19, 11 (Nov. 2009), 1588–1597. https://doi.org/10.1109/TCSVT.2009.2031462

Donghyeon Cho, Jinsun Park, Tae-Hyun Oh, Yu-Wing Tai, and In So Kweon. 2017. Weakly- and Self-Supervised Learning for Content-Aware Deep Image Retargeting. In *2017 IEEE International Conference on Computer Vision (ICCV)*. IEEE. https://doi.org/10.1109/iccv.2017.488

Sabine Coquillart. 1990. Extended Free-Form Deformation: A Sculpturing Tool for 3D Geometric Modeling. In *Proceedings of the 17th Annual Conference on Computer Graphics and Interactive Techniques* (Dallas, TX, USA) *(SIGGRAPH '90)*. Association for Computing Machinery, New York, NY, USA, 187–196. https://doi.org/10.1145/97879.97900

Dongfeng Han, X. Wu, and M. Sonka. 2009. Optimal multiple surfaces searching for video/image resizing - a graph-theoretic approach. In *2009 IEEE 12th International Conference on Computer Vision*. 1026–1033. https://doi.org/10.1109/ICCV.2009.5459380

Raanan Fattal and Dani Lischinski. 2004. Target-driven Smoke Animation. *ACM Trans. Graph.* 23, 3 (Aug. 2004), 441–448. https://doi.org/10.1145/1015706.1015743

Ronald Fedkiw, Jos Stam, and Henrik Wann Jensen. 2001. Visual Simulation of Smoke. In *Proceedings of the 28th Annual Conference on Computer Graphics and Interactive Techniques (SIGGRAPH '01)*. Association for Computing Machinery, New York, NY, USA, 15–22. https://doi.org/10.1145/383259.383260

Sean Flynn, Parris Egbert, Seth Holladay, and Bryan Morse. 2019. Fluid Carving: Intelligent Resizing for Fluid Simulation Data. *ACM Trans. Graph.* 38, 6, Article 238 (Nov. 2019), 14 pages. https://doi.org/10.1145/3355089.3356572

Zahra Forootaninia and Rahul Narain. 2020. Frequency-Domain Smoke Guiding. *ACM Trans. Graph.* 39, 6, Article 172 (Nov. 2020), 10 pages. https://doi.org/10.1145/3414685.3417842

Nick Foster and Ronald Fedkiw. 2001. Practical Animation of Liquids. In *Proceedings of the 28th Annual Conference on Computer Graphics and Interactive Techniques (SIGGRAPH '01)*. ACM, New York, NY, USA, 23–30. https://doi.org/10.1145/383259.383261

R. Furuta, I. Tsubaki, and T. Yamasaki. 2018. Fast Volume Seam Carving With Multipass Dynamic Programming. *IEEE Transactions on Circuits and Systems for Video Technology* 28, 5 (2018), 1087–1101. https://doi.org/10.1109/TCSVT.2016.2620563

Matthias Grundmann, Vivek Kwatra, Mei Han, and Irfan Essa. 2010. Discontinuous Seam-Carving for Video Retargeting. *IEEE CVPR* (2010).

Mark Harris. 2005. Fast Fluid Dynamics Simulation on the GPU. In *ACM SIGGRAPH 2005 Courses* (Los Angeles, California) *(SIGGRAPH '05)*. ACM, New York, NY, USA, Article 220. https://doi.org/10.1145/1198555.1198790

Hsi-Chin Hsin, Tze-Yun Sung, and Chin-Wei Su. 2014. A Fast Wavelet-Based Seam Carving Algorithm for Image Resizing. *Asian Journal of Computer and Information Systems* 2, 5 (Oct. 2014). https://ajouronline.com/index.php/AJCIS/article/view/1796

Nobukatsu Kajiura, Satoshi Kosugi, Xueting Wang, and Toshihiko Yamasaki. 2020. Self-Play Reinforcement Learning for Fast Image Retargeting. In *Proceedings of the 28th ACM International Conference on Multimedia* (Seattle, WA, USA) *(MM '20)*. Association for Computing Machinery, New York, NY, USA, 1755–1763. https://doi.org/10.1145/3394171.3413857

Byungsoo Kim, Vinicius C. Azevedo, Markus Gross, and Barbara Solenthaler. 2019a. Transport-Based Neural Style Transfer for Smoke Simulations. *ACM Trans. Graph.* 38, 6, Article 188 (Nov. 2019), 11 pages. https://doi.org/10.1145/3355089.3356560

Byungsoo Kim, Vinicius C. Azevedo, Markus Gross, and Barbara Solenthaler. 2020. Lagrangian Neural Style Transfer for Fluids. *ACM Transactions on Graphics* 39, 4, Article 52 (2020), 10 pages. https://doi.org/10.1145/3386569.3392473

Byungsoo Kim, Vinicius C. Azevedo, Nils Thuerey, Theodore Kim, Markus Gross, and Barbara Solenthaler. 2019b. Deep Fluids: A Generative Network for Parameterized Fluid Simulations. *Computer Graphics Forum (Proc. Eurographics)* 38, 2 (2019).

Ikjoon Kim, Jidong Zhai, Yan Li, and Wenguang Chen. 2015. Optimizing seam carving on multi-GPU systems for real-time content-aware image resizing. *J. Supercomput.* 71, 9 (2015), 3500–3524. https://doi.org/10.1007/s11227-015-1446-4

Theodore Kim and John Delaney. 2013. Subspace Fluid Re-simulation. *ACM Trans. Graph.* 32, 4, Article 62 (July 2013), 9 pages. https://doi.org/10.1145/2461912.2461987

Tsung-Yi Lin, Michael Maire, Serge Belongie, James Hays, Pietro Perona, Deva Ramanan, Piotr Dollár, and C. Lawrence Zitnick. 2014. Microsoft COCO: Common Objects in Context. In *Computer Vision – ECCV 2014*, David Fleet, Tomas Pajdla, Bernt Schiele, and Tinne Tuytelaars (Eds.). Springer International Publishing, Cham, 740–755.

Frank Losasso, Frédéric Gibou, and Ron Fedkiw. 2004. Simulating Water and Smoke with an Octree Data Structure. In *ACM SIGGRAPH 2004 Papers* (Los Angeles, California) *(SIGGRAPH '04)*. ACM, New York, NY, USA, 457–462. https://doi.org/10.1145/1186562.1015745

A. McAdams, E. Sifakis, and J. Teran. 2010. A Parallel Multigrid Poisson Solver for Fluids Simulation on Large Grids. In *Proceedings of the 2010 ACM SIGGRAPH/Eurographics Symposium on Computer Animation* (Madrid, Spain) *(SCA '10)*. Eurographics Association, Goslar Germany, Germany, 65–74. http://dl.acm.org/citation.cfm?id=1921427.1921438

Antoine McNamara, Adrien Treuille, Zoran Popović, and Jos Stam. 2004. Fluid Control Using the Adjoint Method. *ACM Trans. Graph.* 23, 3 (Aug. 2004), 449–456. https://doi.org/10.1145/1015706.1015744

Ken Museth. 2013. VDB: High-Resolution Sparse Volumes with Dynamic Topology. *ACM Trans. Graph.* 32, 3, Article 27 (July 2013), 22 pages. https://doi.org/10.1145/2487228.2487235

Ken Museth, David E. Breen, Ross T. Whitaker, and Alan H. Barr. 2002. Level Set Surface Editing Operators. *ACM Trans. Graph.* 21, 3 (July 2002), 330–338. https://doi.org/10.1145/566654.566585

Rahul Narain, Jason Sewall, Mark Carlson, and Ming C. Lin. 2008. Fast Animation of Turbulence Using Energy Transport and Procedural Synthesis. In *ACM SIGGRAPH Asia 2008 Papers* (Singapore) *(SIGGRAPH Asia '08)*. ACM, New York, NY, USA, Article 166, 8 pages. https://doi.org/10.1145/1457515.1409119

Andrew Nealen, Takeo Igarashi, Olga Sorkine, and Marc Alexa. 2007. FiberMesh: Designing Freeform Surfaces with 3D Curves. *ACM Trans. Graph.* 26, 3 (July 2007), 41–es. https://doi.org/10.1145/1276377.1276429

Duc Quang Nguyen, Ronald Fedkiw, and Henrik Wann Jensen. 2002. Physically Based Modeling and Animation of Fire. *ACM Trans. Graph.* 21, 3 (July 2002), 721–728. https://doi.org/10.1145/566654.566643

Michael B. Nielsen and Robert Bridson. 2011. Guide Shapes for High Resolution Naturalistic Liquid Simulation. In *ACM SIGGRAPH 2011 Papers* (Vancouver, British Columbia, Canada) *(SIGGRAPH '11)*. ACM, New York, NY, USA, Article 83, 8 pages. https://doi.org/10.1145/1964921.1964978

Zherong Pan, Jin Huang, Yiying Tong, Changxi Zheng, and Hujun Bao. 2013. Interactive Localized Liquid Motion Editing. *ACM Trans. Graph.* 32, 6, Article 184 (Nov. 2013), 10 pages. https://doi.org/10.1145/2508363.2508429

N. Rasmussen, D. Enright, D. Nguyen, S. Marino, N. Sumner, W. Geiger, S. Hoon, and R. Fedkiw. 2004. Directable Photorealistic Liquids. In *Proceedings of the 2004 ACM SIGGRAPH/Eurographics Symposium on Computer Animation* (Grenoble, France) *(SCA '04)*. Eurographics Association, Goslar Germany, Germany, 193–202. https://doi.org/10.1145/1028523.1028549

Karthik Raveendran, Chris Wojtan, Nils Thuerey, and Greg Turk. 2014. Blending Liquids. *ACM Trans. Graph.* 33, 4, Article 137 (July 2014), 10 pages. https://doi.org/10.1145/2601097.2601126

Michael Rubinstein, Ariel Shamir, and Shai Avidan. 2008. Improved Seam Carving for Video Retargeting. *ACM Trans. Graph.* 27, 3, Article 16 (Aug. 2008), 9 pages. https://doi.org/10.1145/1360612.1360615

Michael Rubinstein, Ariel Shamir, and Shai Avidan. 2009. Multi-operator Media Retargeting. In *ACM SIGGRAPH 2009 Papers* (New Orleans, Louisiana) *(SIGGRAPH '09)*. ACM, New York, NY, USA, Article 23, 11 pages. https://doi.org/10.1145/1576246.1531329

Syuhei Sato, Yoshinori Dobashi, Theodore Kim, and Tomoyuki Nishita. 2018b. Example-Based Turbulence Style Transfer. *ACM Trans. Graph.* 37, 4, Article 84 (July 2018), 9 pages. https://doi.org/10.1145/3197517.3201398

Syuhei Sato, Yoshinori Dobashi, and Tomoyuki Nishita. 2018a. Editing Fluid Animation Using Flow Interpolation. *ACM Trans. Graph.* 37, 5, Article 173 (Sept. 2018), 12 pages. https://doi.org/10.1145/3213771

Arnaud Schoentgen, Pierre Poulin, Emmanuelle Darles, and Philippe Meseure. 2020. *Particle-Based Liquid Control Using Animation Templates.* Eurographics Association, Goslar, DEU. https://doi.org/10.1111/cgf.14103

Thomas W. Sederberg and Scott R. Parry. 1986. Free-Form Deformation of Solid Geometric Models. In *Proceedings of the 13th Annual Conference on Computer Graphics and Interactive Techniques (SIGGRAPH '86).* Association for Computing Machinery, New York, NY, USA, 151–160. https://doi.org/10.1145/15922.15903

Z. K. Senturk and D. Akgun. 2019. Seam Carving Based Image Retargeting: A Survey. In *2019 1st International Informatics and Software Engineering Conference (UBMYK).* 1–6. https://doi.org/10.1109/UBMYK48245.2019.8965618

Rajsekhar Setaluri, Mridul Aanjaneya, Sean Bauer, and Eftychios Sifakis. 2014. SPGrid: A Sparse Paged Grid Structure Applied to Adaptive Smoke Simulation. *ACM Trans. Graph.* 33, 6, Article 205 (Nov. 2014), 12 pages. https://doi.org/10.1145/2661229.2661269

Tamar Rott Shaham, Tali Dekel, and Tomer Michaeli. 2019. SinGAN: Learning a Generative Model From a Single Natural Image. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV).*

Lin Shi and Yizhou Yu. 2005. Taming Liquids for Rapidly Changing Targets. In *Proceedings of the 2005 ACM SIGGRAPH/Eurographics Symposium on Computer Animation* (Los Angeles, California) *(SCA '05).* ACM, New York, NY, USA, 229–236. https://doi.org/10.1145/1073368.1073401

Assaf Shocher, Shai Bagon, Phillip Isola, and Michal Irani. 2019. InGAN: Capturing and Retargeting the "DNA" of a Natural Image. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV).*

Denis Simakov, Yaron Caspi, Eli Shechtman, and Michal Irani. 2008. Summarizing visual data using bidirectional similarity. *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition*, 1 – 8. https://doi.org/10.1109/CVPR.2008.4587842

Barry F. Smith, Petter E. Bjørstad, and William D. Gropp. 1996. *Domain Decomposition: Parallel Multilevel Methods for Elliptic Partial Differential Equations.* Cambridge University Press, New York, NY, USA.

O. Sorkine, D. Cohen-Or, Y. Lipman, M. Alexa, C. Rössl, and H.-P. Seidel. 2004. Laplacian Surface Editing. In *Proceedings of the 2004 Eurographics/ACM SIGGRAPH Symposium on Geometry Processing* (Nice, France) *(SGP '04).* Association for Computing Machinery, New York, NY, USA, 175–184. https://doi.org/10.1145/1057432.1057456

Jos Stam. 1999. Stable Fluids. In *Proceedings of the 26th Annual Conference on Computer Graphics and Interactive Techniques (SIGGRAPH '99).* ACM Press/Addison-Wesley Publishing Co., New York, NY, USA, 121–128. https://doi.org/10.1145/311535.311548

Weimin Tan, Bo Yan, Chuming Lin, and Xuejing Niu. 2019. Cycle-IR: Deep Cyclic Image Retargeting. *IEEE Transactions on Multimedia* PP (12 2019), 1–1. https://doi.org/10.1109/TMM.2019.2959925

Nils Thuerey. 2016. Interpolations of Smoke and Liquid Simulations. *ACM Trans. Graph.* 36, 1, Article 3 (Sept. 2016), 16 pages. https://doi.org/10.1145/2956233

N. Thürey, R. Keiser, M. Pauly, and U. Rüde. 2006. Detail-preserving Fluid Control. In *Proceedings of the 2006 ACM SIGGRAPH/Eurographics Symposium on Computer Animation* (Vienna, Austria) *(SCA '06).* Eurographics Association, Aire-la-Ville, Switzerland, Switzerland, 7–12. http://dl.acm.org/citation.cfm?id=1218064.1218066

Adrien Treuille, Antoine McNamara, Zoran Popović, and Jos Stam. 2003. Keyframe Control of Smoke Simulations. *ACM Trans. Graph.* 22, 3 (July 2003), 716–723. https://doi.org/10.1145/882262.882337

B. Wang, H. Xiong, Z. Ren, and C. W. Chen. 2014. Deformable Shape Preserving Video Retargeting With Salient Curve Matching. *IEEE Journal on Emerging and Selected Topics in Circuits and Systems* 4, 1 (2014), 82–94. https://doi.org/10.1109/JETCAS.2014.2298313

Martin Wicke, Matt Stanton, and Adrien Treuille. 2009. Modular Bases for Fluid Dynamics. *ACM Trans. Graph.* 28, 3, Article 39 (July 2009), 8 pages. https://doi.org/10.1145/1531326.1531345

Yuwei Xiao, Szeyu Chan, Siqi Wang, Bo Zhu, and Xubo Yang. 2020. An Adaptive Staggered-Tilted Grid for Incompressible Flow Simulation. *ACM Trans. Graph.* 39, 6, Article 171 (Nov. 2020), 15 pages. https://doi.org/10.1145/3414685.3417837

Jong-Chul Yoon, Sun-Young Lee, In-Kwon Lee, and Henry Kang. 2014. Optimized image resizing using flow-guided seam carving and an interactive genetic algorithm. *Multim. Tools Appl.* 71, 3 (2014), 1013–1031. https://doi.org/10.1007/s11042-012-1242-6

Yizhou Yu, Kun Zhou, Dong Xu, Xiaohan Shi, Hujun Bao, Baining Guo, and Heung-Yeung Shum. 2004. Mesh Editing with Poisson-Based Gradient Field Manipulation. *ACM Trans. Graph.* 23, 3 (Aug. 2004), 644–651. https://doi.org/10.1145/1015706.1015774

Ya Zhou, Zhibo Chen, and Weiping Li. 2021. Weakly Supervised Reinforced Multi-Operator Image Retargeting. *IEEE Transactions on Circuits and Systems for Video Technology* 31, 1 (2021), 126–139. https://doi.org/10.1109/TCSVT.2020.2977943

Yongning Zhu and Robert Bridson. 2005. Animating Sand As a Fluid. *ACM Trans. Graph.* 24, 3 (July 2005), 965–972. https://doi.org/10.1145/1073204.1073298