DOI: 10.1111/cgf.14342
Eurographics Symposium on Rendering 2021
A. Bousseau and M. McGuire
(Guest Editors)

Volume 40 (2021), Number 4

Video-Based Rendering of Dynamic Stationary Environments from Unsynchronized Inputs

T. Thonat[1,2], Y. Aksoy[3], M. Aittala[4,5], S. Paris [2], F. Durand[4], and G. Drettakis[1]

[1]Inria, Université Côte d'Azur, [2]Adobe Research, [3]ETH Zürich, [4]MIT CSAIL, [5]NVIDIA

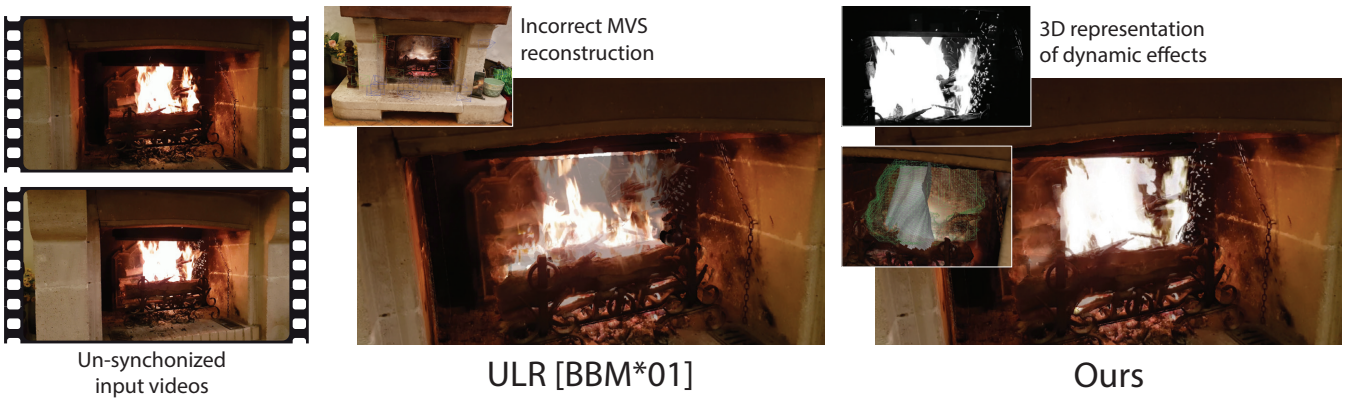

Figure 1: *Unsynchronized video capture (left) is very challenging for Multi-View Stereo (MVS) and Image-Based Rendering techniques that use the resulting geometry (middle inset), where the flames have not been reconstructed. Our method enables free-viewpoint navigation in a Video-Based Rendering context for dynamic stationary effects such as fire. Our method extracts a 3D representation of dynamic effects as well as alpha-matte videos, that we use in our seamless spatio-temporal video blending scheme (left inset). When reprojecting videos onto the MVS geometry, using for example the weighting scheme of [BBM*01] (middle), the flames are incorrectly projected onto the fireplace wall. Our method (right) preserves the volumetric and semi-transparent nature of the fire during free-viewpoint rendering. These dynamic effects are best viewed in the supplemental videos.*

Abstract

Image-Based Rendering allows users to easily capture a scene using a single camera and then navigate freely with realistic results. However, the resulting renderings are completely static, and dynamic effects – such as fire, waterfalls or small waves – cannot be reproduced. We tackle the challenging problem of enabling free-viewpoint navigation including such stationary dynamic effects, but still maintaining the simplicity of casual capture. Using a single camera – instead of previous complex synchronized multi-camera setups – means that we have unsynchronized videos of the dynamic effect from multiple views, making it hard to blend them when synthesizing novel views. We present a solution that allows smooth free-viewpoint video-based rendering (VBR) of such scenes using temporal Laplacian pyramid decomposition video, enabling spatio-temporal blending. For effects such as fire and waterfalls, that are semi-transparent and occupy 3D space, we first estimate their spatial volume. This allows us to create per-video geometries and alpha-matte videos that we can blend using our frequency-dependent method. We also extend Laplacian blending to the temporal dimension to remove additional temporal seams. We show results on scenes containing fire, waterfalls or rippling waves at the seaside, bringing these scenes to life.

Keywords: Video-based rendering

1. Introduction

Recent Image-Based Rendering (IBR) algorithms provide high-quality, free-viewpoint navigation of entire scenes captured using only a set of photos as input [PZ17, HPP*18, RK20]. One major advantage is that these methods only require photos taken with a *single* camera, with sufficient density to allow successful camera calibration and multi-view stereo (MVS) reconstruction [SF16, SZPF16, Rea18]. However, they suffer from a major limi-

tation: all content is static and lifeless. The dynamic nature of environments like the waves rippling on a beach or the flames in a fireplace is not handled by previous methods, despite specific multi-camera performance capture solutions e.g., [CTMS03, CCS*15], that allow moving humans to be inserted in IBR scenes. These dynamic elements are integral to the nature of these scenes and easily capturing them *in situ* is beyond the ability of existing IBR techniques.

We introduce a video-based rendering method for free-viewpoint navigation of scenes with stationary *stochastic* dynamic phenomena such as waterfalls, streams, small waves, or fire. Our method retains the advantage of practical capture since it only requires a single smartphone and a cheap tripod, a handful of videos of the scene and the same number of wide-baseline photos as required for Structure from Motion (SfM) and MVS. This is in contrast to previous video-based rendering solutions that mostly involve complex *synchronized multi-camera* systems and typically focused on human motion [CTMS03, CCS*15, WBR07, LSS*19] or only allow limited *transitions* between views [BBPP10, TKKT12].

The main challenge for our casual capture setup comes from the fact that we blend *unsynchronized* videos from different viewpoints, thus the different views of the dynamic phenomenon are not photo-consistent. This raises two important issues.

First, naive blending between input videos in the novel view is likely to cause excessive blurring and generate unsightly temporal discontinuities, due to the unsynchronized multi-view nature of our data and the fact that we loop short input sequences over time. Our main insight is the use of Laplacian decomposition to manipulate different temporal and spatial frequencies separately while preserving their original statistics. This approach provides smooth transitions for free-viewpoint navigation in space and time. Our method provides the first rendering solution allowing free-viewpoint navigation in a scene with dynamic stationary elements, significantly reducing discontinuities both for looping and blending transitions between different cameras.

Second, the phenomena we treat can be volumetric and semi-transparent (fire, waterfalls etc.); the unsynchronized nature of the videos precludes the use of SfM/MVS, and tomography-style reconstruction (e.g., [GKHH12]) would fail since opacity and the resulting volume cannot be reliably estimated. Instead, we present a solution that localizes the effect in 3D space by providing approximate per-view geometry for reprojection, and in 2D space by estimating per-view alpha mattes.

Our Laplacian blending, per-view geometry and alpha mattes allow free-viewpoint rendering with stochastic dynamic effects captured with unsynchronized videos. We show results (Figure 17, Figure 15 and supplemental material) on phenomena such as fire, waterfalls or rippling waves in a seaside scene, bringing these scenes to life and allowing some level of editability, e.g., adjusting the "liveliness" of motion. Our method allows interactive free-viewpoint navigation while maintaining the high-frequency visual components of these effects.

2. Related Work

Image-based rendering (IBR) algorithms (e.g., [BBM*01, HPP*18, RK20]) only require a sparse set of unstructured photos of a scene to provide high quality, free-viewpoint rendering, often in real-time. Each such method makes some trade-off between exploration capabilities, ease of capture and complexity of treated scenes. Similarly, our method strives to allow free-viewpoint navigation with dynamic content, thus "bringing the captured scene to life"; inevitably we are also confronted with various trade-offs.

Many IBR methods (e.g., [BBM*01, CDSHD13, HRDB16]) rely on a geometric representation of the scene that is used to reproject the input photos onto the novel view, and assume photo-consistency between input images. Recent IBR methods have attempted to overcome inaccuracies in reconstructed geometry using per-view representations [CDSHD13, HRDB16], learned blending weights [HPP*18] or modeling uncertainty via a volumetric representation [PZ17]. In our context, the photo-consistency assumption is broken since we work with unsynchronized videos with events corresponding to different moments in time with possibly different appearance. However, our technical choices are often inspired by those developed for IBR, e.g., per-view geometry or volumetric representations used for the dynamic effects we model.

More recently, *neural rendering* [TFT*20] has allowed great advances in IBR quality. However, some recent methods focus more on single objects or restricted scene sizes [MST*20], and thus have difficulty with the kind of scenes we target. This restriction was observed in recent methods [RK20, ZRSK20] that can handle large scenes, but are also limited to static content. Neural Sparse Voxel Fields [LGZL*20, PCPMMN21] can treat dynamic sequences, but only for datasets captured with synchronized cameras. We discuss multi-video neural rendering below (Sec. 2.3).

Several methods have tackled the problem of capturing humans [CTMS03, CCS*15, WBR07, SH07], typically using multiple synchronized videos. These methods often use human body based priors such as tracking of specific human parts or global template fitting. In contrast, our focus is on single-camera casual capture, and on stochastic phenomena such as water or fire.

2.1. Video looping and compositing

Videos are a natural way to capture the dynamic nature of a scene. However, a video only provides a fixed segment in both time and space. Therefore, many methods have been proposed to overcome these limitations by creating endless video streams and combining the information from multiples videos.

Video Textures [SSSE00] create an infinitely long non-repeating video stream from one static video clip. By finding probabilities of transitions between frames, the method produces a smart random video player for a wide variety of motions. Panoramic Video Textures [AZP*05] use a rotating camera to capture dynamic effects with a wider view angle than a single video. This method generates dynamic effects for the entire panorama, but with a fixed viewpoint. Bai et al. [BAAR12] allow users to selectively remove large scale motion in a video, creating cinemagraphs, i.e., photos with some

subset having a looping motion. Liao et al. [LJH13] automatically capture different levels of dynamic behavior in a given static video, enabling a whole spectrum of video loops. More recently, He et al. [HLSH17] create single gigapixel panorama video loops from a structured but unsynchronized capture of videos, with detailed rendering and a wider range of motions. These purely video-space looping methods allow the capture of virtually any kind of looping motion, without severe limitations on the content. However they either have a fixed field of view, or rely on complex and computationally costly graphcut optimizations. As we shall see later (Sec. 4.1.2), we create loops of our stochastic content using a simpler and cheaper solution with satisfactory results.

2.2. Exploring video datasets

Taneja et al. [TBP10] model dynamic elements in a scene using synchronized moving cameras, using reprojection on a static reconstructed background to identify ghosting introduced by the dynamic foreground. Ballan et al. [BBPP10] use a sparse, unstructured but synchronized video capture setup, based on adjusting transition points to navigate around a performer. Videoscapes [TKKT12] rely on a huge number of wide-baseline unstructured and unsynchronized videos to explore a scene, by identifying transitions between moving video clips. The two last methods try to reduce the impact of transitions between clips. In contrast, we focus on free-viewpoint navigation rather than transitions, and achieve this by limiting the scope of scenes we treat.

Interactive Viewpoint Textures [LTK12] use a semi-structured and unsynchronized video capture setup, allowing free viewpoint navigation along a path. This method depends on optical flow that cannot be successfully computed on our wide-baseline, unsynchronized video input.

2.3. Multi-video dense reconstruction

Several methods overcome the challenges of dynamic scenes by using dense or constrained capture setups. The method of Gregson et al. [GKHH12] uses a dense capture setup of synchronized videos to reconstruct turbulent fluids. It uses visible light computed tomography and is able to render the captured fluids without explicit 3D volumes. Zang et al. [ZIT*18] use only one sensor but capture thousands of X-ray projections, showing high quality reconstructions of solid objects degrading over time. Okabe et al. [OAO12] extract a fluid from a monocular video by introducing a matting technique, first estimating the background followed by an initial alpha matte, subsequently refined using gradient domain processing. We also refine an alpha matte, but the conditions for background extraction are different (see also Sec. 5.4). In later work, the same authors [ODAO15] use a sparse setup of synchronized videos to synthesize dynamic effects, in terms of both appearance and volume, instead of relying on reconstruction. More recent solutions employ deep learning to estimate opacity, again in a synchronized multi-video context [LSS*19, ZIW*20]. Such tomography-style solutions cannot be used in our context, since the unsynchronized nature of our input videos does not allow opacity estimation required to recover the volume of the dynamic effect. Instead, we disassociate the computations, i.e., we independently compute an approximate geometry representation for the dynamic effect and estimate alpha mattes.

Recent work in video-based neural rendering [TFT*20] provides solutions with impressive visual quality, e.g., the immersive light field video [BFO*20]; however it needs 16 to 48 synchronized cameras. Other learning-based solutions apply to specific capture setups, such as a liquid refractive surface [TLY20]. Finally, X-fields [MB20] can handle limited dynamic temporal phenomena, but is restricted to small baseline capture typical to light-fields. Deep learning methods are a very promising avenue for future work; current solutions however do not address the wide-baseline, unsynchronized capture of dynamic phenomena that we target.

3. Overview

Before addressing the actual compositing and rendering algorithm, we briefly discuss our capture procedure, which combines unstructured photo capture with significant coverage of the scene (i.e. between 50 and 100 photos), and sparse video capture using a tripod, focusing on the dynamic elements of the scene (between 5 and 10 videos). We used a phone camera for videos, and for photos. Videos were recorded at 30 fps with 2K resolution, and photos with 4K resolution. The input videos need to be sufficiently spaced to construct visual hulls of the dynamic effect. In our scenes, the views have an angular deviation ranging from 10 to 30 degrees.

A median image is first extracted from each input video. We reconstruct the overall scene geometry, using an off-the-shelf SfM and MVS algorithm [Rea18], taking the photos and median images as input. For effects well localized in space, such as the fireplace, our solution uses a volumetric carving approach to recover localized approximate geometry as described in section 5. Water planes, or any other "scene surrounding" effect, prevent useful carving and thus required manual intervention to obtain a ground plane [†]. After this step, the input of our method is a set of calibrated cameras corresponding to both the input videos and the input photos, as well as a reasonable approximate geometry for the static scene parts. An overview of our solution is shown in Figure 2.

In the following section (section 4), we start with the simplest case of an opaque surface such as a (muddy water) water basin with rippling waves, where we assume known geometry and location of the dynamic effect. There are two kinds of spatio-temporal discontinuities in novel views, those due to the non-looping nature of input videos, and those due to the lack of synchronization between cameras. We introduce a spatio-temporal blending approach, by extending Laplacian blending, and propose a simple yet effective solution for video looping (Figure 2, green boxes).

For more complex cases such as a waterfall or flames, the dynamic effect is localized in 3D space in a potentially time-varying manner. To allow reprojection into a novel view, we must estimate a 3D supporting geometry, and since these cases often involve semi-transparent elements, we need to develop an effective matting strategy. We introduce a solution to motion localization using frequency-based analysis (section 5), and compute a voxelized visual hull of the dynamic effects (blue box in Figure 2). The voxels are used to extract per-view geometric proxies, allowing reprojection of the input videos into the novel view.

[†] Scenes *Beach*, *Seaside* and *Cave*

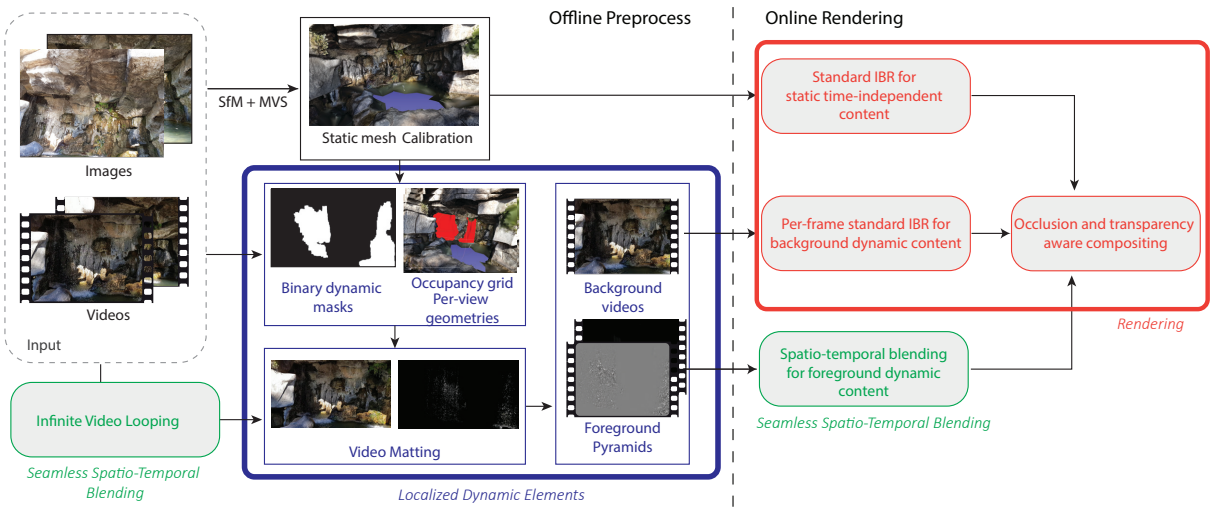

Figure 2: ***Method overview.*** *Using a multi-view dataset composed of unstructured images and unsynchronized videos, we first perform standard SfM and dense reconstruction. In a preprocessing step we create loops of the input videos, then compute a 3D representation of the dynamic phenomenon if necessary. Spatio-temporal blending of the unsynchronized videos is combined with other image-based rendering steps for final rendering of the background. The spatio-temporal looping and blending of the foreground dynamic elements are indicated in green and are described in Sec. 4; the 3D localization of dynamic elements are indicated in blue (Sec. 5) and the rendering steps for both foreground and background are in red (Sec. 6).*

4. Seamless Spatio-temporal Blending

We seek remove the two kinds of spatio-temporal discontinuities discussed above by making these boundaries seamless, thus avoiding unpleasant visual discontinuities in the final free-viewpoint navigation output.

To do this, we extend and adapt Laplacian blending techniques to the spatio-temporal context of *unsychronized* multi-view videos, and specifically free-viewpoint Video-Based Rendering (VBR).

In this section, we introduce the background and the basic principles of our temporal Laplacian pyramid. For now, we assume the simplest case, where the 3D geometry of the scene, and notably the dynamic part, is known. This assumption allows to reproject video content from input viewpoints onto any novel view. In Section 5, we lift these restrictions and discuss how to localize dynamic phenomena in space, allowing us to treat cases such as fire and waterfalls that occupy a volume and are semi-transparent.

Because no single input sequence sees the whole scene, we create the novel view output by stitching several input videos together from different locations, thereby creating spatial discontinuities or boundaries. Since users freely control the output viewpoint, these boundaries change over time. In addition, to allow users to explore the scene without a time limit, we generate video loops from our input sequences that last around and 2 s. One solution is to make each sequence loop, which introduces a temporal boundary after each repetition.

From this perspective, the output video generated by our approach is a patchwork made of the input sequences with complex space-time boundaries separating them. Next, we explain how we produce a seamless composite.

4.1. Seamless Compositing

Reprojection-based IBR methods [BBM*01, EDDM*08, HPP*18] use *blending fields* to reduce artifacts due to spatial discontinuities. Blending fields represent how much each input view contributes to the novel view and change smoothly with camera motion. However, in the case of unsynchronized videos, such blending fields produce additional temporal discontinuities that must be accounted for. Our insight is that the amount of discrepancy between the input videos depends on the temporal frequency of the content. Our goal is to have a video representation that identifies this frequency-dependent content and which adapts blending based on this.

We introduce a new compositing algorithm by extending and adapting the multi-scale blending approach of Burt and Adelson [BA83]. For completeness we briefly review this technique in Appendix A; we next show how to adapt this pyramidal construction to our context.

Intuitively, this multi-scale operation blends each frequency band at "the right scale", i.e., high frequencies are blended over a very short distance, thereby avoiding potential ghosting artifacts, while low frequencies are slowly cross-faded to prevent the introduction of unsightly discontinuities.

We introduce an extension to this method using video volumes, considering time as the main dimension instead of space. We use the kernel $\frac{1}{16}[1\ 4\ 6\ 4\ 1]$ along the time axis for convolutions in the reduce and expand operations (see supplemental). Each level of the temporal Laplacian video pyramid is itself a video representing a temporal frequency band of the input video. However, instead of keeping levels of different length, we use the expanded pyramid equivalent representation, where we apply the 1D expand operator [BA83] to upscale each level in time up to the input length. Such

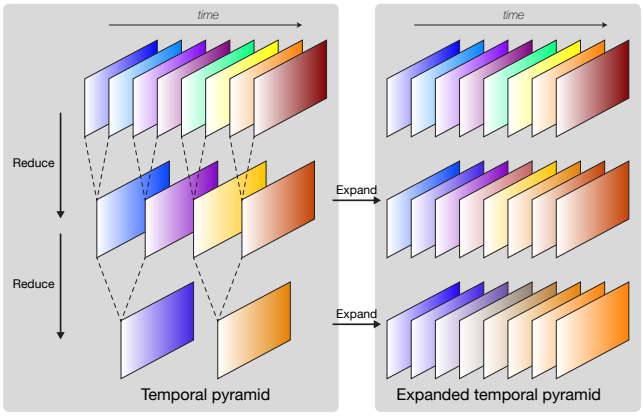

Figure 3: ***Expanded Temporal Laplacian Pyramid.*** *Given an input video, a pyramid is computed with each level having half the number of frames than the previous level (left). Each level is then expanded to have the same length as the input (right). Figure 5 shows an example of such a decomposition.*

a scheme allows a better visualization of the contribution of each level since the input video simply becomes the per-frame per-pixel sum of all the levels (Fig. 3 and 4). With this approach, we preserve the frequency-domain statistics of the scene, e.g., we do not introduce temporal discontinuities nor smooth out existing temporal phenomena like ripples in waves. Unlike the method of Burt and Adelson that applies to images and is purely spatial [BA83], ours works on videos and seeks to preserve temporal statistics.

4.1.1. Spatio-temporal Laplacian blending

A direct extension of IBR to videos would consider input videos as time-dependent textures. At each new frame, the input images would be updated and used by any IBR method. However, since the videos are not synchronized, the images are not photo-consistent. As a result, a weighted average produces blurred results, losing the spatial high frequency details of the different inputs. On the other hand, IBR with a sharp blending field would produce image-space seams on content that is not photo-consistent. These seams would be particularly visible when moving the viewpoint. View-dependent image stitching (e.g., using Poisson compositing [PGB03] or standard spatial Laplacian blending [BA83]) methods produce similar results.

Since our videos depict stochastic effects that are stationary in time, photo-consistency heavily depends on the temporal frequency we are considering. Low frequencies correspond effectively to the temporal average of each video, which does not vary much from one viewpoint to another as for example a fireplace would stay overall at the same location. In contrast, temporal high frequencies – such as flames – are highly uncorrelated between frames and viewpoints, but they share a similar spatial distribution across viewpoints.

To support free-viewpoint navigation, we introduce a spatio-temporal blending strategy, corresponding to multiple IBR blending schemes, that are independent per temporal frequency. The key idea is that each different temporal frequency band represents video content that we blend using appropriate IBR blending strategies. In Figure 3, we see that each input video V_i is decomposed, using a temporal Laplacian Pyramid, into n_L video-levels $\mathcal{K}[V_i]_\ell$ such that for each frame t:

$$V_i(t) = \sum_{\ell=0}^{n_\mathrm{L}-1} \mathcal{K}[V_i]_\ell(t) \tag{1}$$

where $\mathcal{K}[V_i]_0$ is a video containing the highest temporal frequency band, while $\mathcal{K}[V_i]_{n_\mathrm{L}-1}$ is the lowest temporal frequency residual, i.e., a video where each frame is an approximation of the video average over time (Figure 3, Figure 4 (right)).

We now describe the blending method for a given frame, dropping the t dependency for convenience. For each video i and each level ℓ, the video level textures $\mathcal{K}[V_i]_\ell$ are projected onto the dynamic effect geometry:

$$\hat{\mathcal{K}}[V_i]_\ell = R(\mathcal{K}[V_i]_\ell) \tag{2}$$

where $R(\cdot)$ is the reprojection operator, with respect to the geometry and the novel view, and $\hat{\mathcal{K}}[V_i]_\ell$ is the reprojected content from level ℓ of video i.

We also compute a per-fragment blending field based on ULR penalties [BBM*01] using the 4 closest video viewpoints. However we apply a different normalization per temporal frequency. Considering $w[V_i]$ the penalty associated to a view i for a given fragment, we use a *softmax* normalization scheme:

$$w[V_i]_\ell = \frac{e^{-\lambda_\ell w[V_i]}}{\sum_i e^{-\lambda_\ell w[V_i]}} \tag{3}$$

where $w[V_i]_\ell$ is the final blending weight associated to view i and level ℓ. The parameter λ_ℓ controls the blending field sharpness. For the low frequencies, a small λ_ℓ produces a smooth blending field, similar to standard ULR. For the high frequencies, a high λ_ℓ produces a piecewise-constant blending field where smooth transitions have small support in image-space; this can be seen in the third row of Figure 5. In all our examples we used $\lambda_\ell = 50 \cdot (n_\mathrm{L} - \ell)$.

These per-frequency blending fields allow us to compute per-frequency blendings F_ℓ, producing the different levels of our output temporal Laplacian pyramid. The final rendered frame F is then obtained by collapsing this pyramid (Figure 5). Since we work with expanded temporal Laplacian pyramids, the collapse is simply the sum of the different levels:

$$F = \sum_{\ell=0}^{n_\mathrm{L}-1} F_\ell = \sum_{\ell=0}^{n_\mathrm{L}-1} \sum_i w[V_i]_\ell \cdot \hat{\mathcal{K}}[V_i]_\ell \tag{4}$$

Considering images as videos with the same image in every frame, their temporal Laplacian pyramid levels $\mathcal{K}[V_i]_\ell$ for $\ell \neq n_\mathrm{L} - 1$ are equal to 0 while the residual level $\mathcal{K}[V_i]_{n_\mathrm{L}-1}$ is equal to the input image, meaning that blending field based IBR methods [BBM*01] can be regarded as a special case of our method. Moreover, if the same blending field is used for all frequencies, our method is equivalent to applying a standard blending field based IBR using videos simply as time-dependent textures.

4.1.2. Seamless Video Looping

A naive option to create video loops is to restart the input sequence when it ends (Fig. 6, middle). This produces strong visual discon-

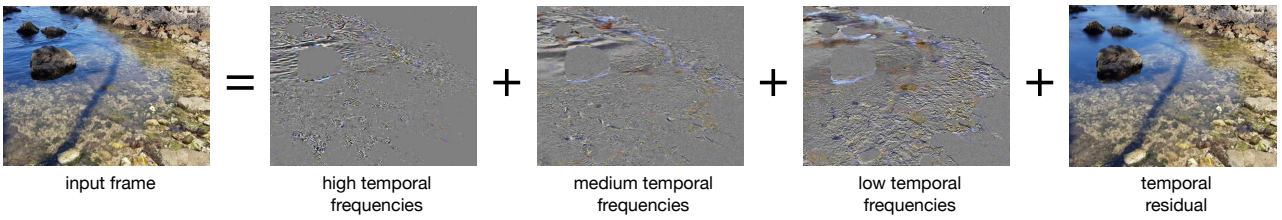

Figure 4: ***Sample frames from an expanded temporal pyramid.*** *Figure 3 illustrates how we generated these frames.*

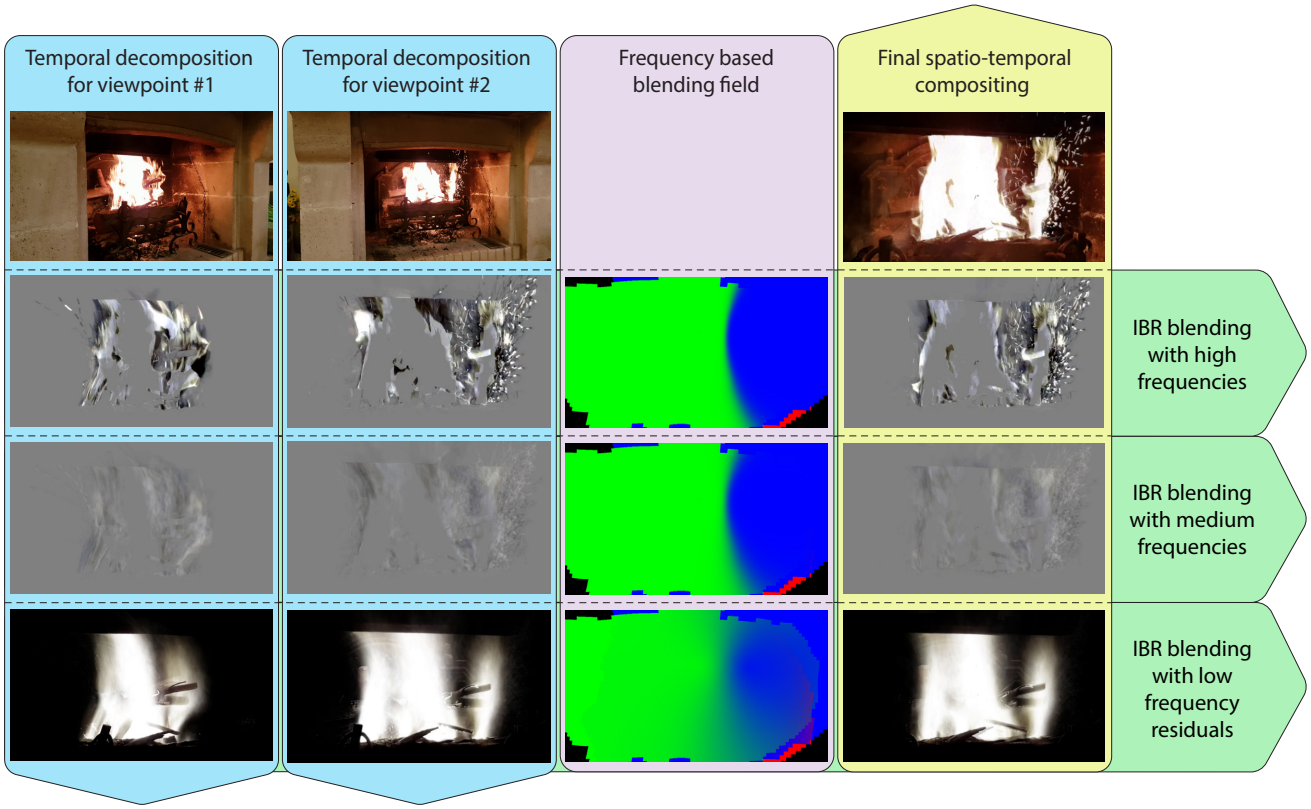

Figure 5: ***Spatio-temporal blending.*** *Left two columns: Reprojection of the input video foreground for different temporal levels (from top to bottom row: input, high frequency band, middle frequency band and low frequency residual). Middle column: A different blending field is computed for each temporal frequency, favoring smooth transition for low frequencies. Right: The final foreground rendering is obtained by collapsing the per-frequency IBR blendings.*

tinuities each time, which is not acceptable in our context. A better approach is to introduce some overlap and cross-fade progressively between the end of the sequence and its beginning. While better than naive looping, this raises the question of how fast to cross-fade: too slow and ghosting artifacts appear, too fast and discontinuities come back. Several options based on optimization have been proposed to address this issue, e.g., [BAAR12, LJH13], and they could possibly work in our context. However, they all come at a significant computational cost. Instead, we make our videos loop while being efficient, by simply using temporal Laplacian blending. Implementation details can be found in the supplemental material. This approach greatly reduces the temporal discontinuities as illustrated in Figure 6. Once looped, our videos are typically $128 = 2^7$ frames long, which corresponds to $n_L = 8$ temporal frequencies.

5. Localized Dynamic Elements

In the previous section, we assumed the dynamic content is located on a reconstructed surface. We now discuss how we can obtain this localization in space as dynamic and especially unsynchronized content is not well reconstructed by multi-view stereo. Typical examples of this are flames or a water fountain, shown in Fig. 1 and 15. In addition, such dynamic phenomena are often semi-transparent, posing an additional challenge since rendering these

Figure 6: ***Video looping.*** *Natural videos played back to back show a strong temporal discontinuity (middle), while our temporal Laplacian blending scheme produces a seamless video loop (right). Slices are represented with time on the vertical axis and space on the horizontal axis. Displayed slices correspond to the zoomed-in segment (left).*

effects will require matting, i.e., separating the foreground from the background.

Figure 7: *Frames from two videos of a fire (corners), which is a dynamic stochastic phenomenon that is localized in 3D. When captured with unsynchronized video and photos, the geometry of the phenomena is not reconstructed (center).*

Recall that we have *unsynchronized* videos of a stochastic, typically semi-transparent, phenomenon localized in space. Previous methods that capture such phenomena have used multi-camera *synchronized* video setups [GKHH12]. In their context, the typical method first estimates volume density (akin to opacity), implicitly estimating the spatial extent of the volume with a tomography-like approach. Such an approach is impossible in our case, since the unsynchronized nature of our videos does not allow estimation of opacity. Instead, we *invert* the order of operations, and first approximate an overall bounding volume of the dynamic phenomenon over time, then provide supporting 3D geometry to allow reprojection of videos and finally estimate per-view per-frame opacity in the form of alpha mattes.

We use an approach akin to constructing a visual hull to estimate the overall volume, since low temporal frequencies are mostly coherent between the views but lack image features. We follow with depth propagation to assign depth to video-specific dynamic content, and then depth-map smoothing to reduce texture distortion during reprojection. This approach allows us to represent semi-transparent regions for phenomena like flames and water streams.

5.1. Extraction of dynamic video regions

We first extract a volumetric representation of the effect that is consistent across views, and identify the regions of the input videos that actually capture a dynamic effect. We extract a per video binary non dynamic segmentation mask M_b by thresholding per-pixel total variation, i.e., the sum over all frames of the L_1 distance between the color at t and $t+1$. A pixel belongs to the mask if its average variation over the video is greater than 5, using 8 bit RGB color values. These masks are a rough estimation of the dynamic pixels in our videos (Fig. 8). We estimate depth in the volumetric representation which we propagate to each view-specific dynamic part, as no matching across views is possible in these regions (see Sec. 5.3).

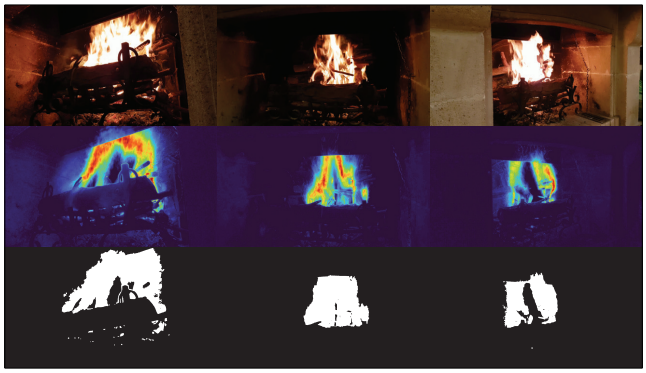

Figure 8: ***Rough binary masks for dynamic effects using total variation****: Three frames from input videos (top) with their corresponding per-pixel total variation (middle) and their corresponding binary masks obtained by thresholding the total variation (bottom). For each video, the binary mask roughly indicates the dynamic effect silhouette from that viewpoint.*

5.2. Voxelized visual hull

Given the dynamic segmentation masks M_b, we combine the information from all input videos to locate the motion in world space. Visual hulls [Lau94, MBR*00], use multi-view silhouette information to recover an approximate 3D shape. We compute a visual hull of our dynamic effects using a voxel grid. For each voxel we randomly sample points in its volume, and reproject them onto the input video segmentation masks. We perform visibility tests using the static geometry, assuming the static parts are opaque. We consider that a voxel belongs to the visual hull if for most of its samples, the input videos seeing the associated point agree on its dynamic nature. More specifically, a voxel has to be visible from at least 4 views, and its reprojections must be part of the dynamic mask in at least 90% of these input views. In all our experiments, we used a 256^3 grid, and we sample 64 points in each voxel.

5.3. Per view geometry

The voxel occupancy grid approximately localizes the dynamic effect in 3D space. During rendering, we need to reproject the textures onto a surface. Since the videos are not synchronized, there is not enough multi-view consistency information to create a single global geometry; this could also pose issues with visibility, etc.

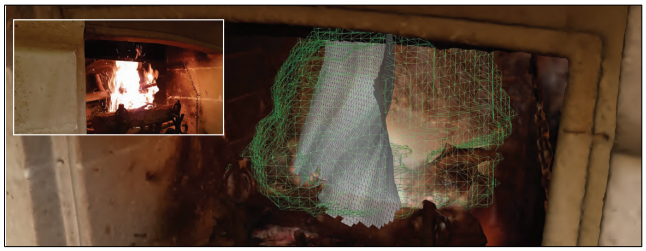

Figure 9: ***Geometric representation.*** *Occupancy grid for the fire scene (in green), with per view geometry (in grey) associated to the view in the inset.*

Figure 10: ***Total variation based depth propagation.*** *For an input video (top left) we extract a median depth along the volumetric representation (top right). However, to assign plausible depth to dynamic video-specific regions not in the volumes, we propagate this depth (bottom right) using the per-pixel total variation as a guide (bottom left).*

We need a geometry that represents the video content without too much distortion and that is able to provide some notion of consistency when switching from one video to another. We thus model the dynamic effects using per-view geometry, that can be seen as approximate view-dependent "billboards" (see Figure 9).

Given an input view and the voxel occupancy grid, we cast a ray for each pixel and gather the depths of the non empty cells traversed by the ray. In order to assign a depth that is robust to temporal variations and sufficiently compatible with the other viewpoints, we extract the median of the per-pixel depth histogram. We then fill holes and remove outliers using morphological operations.

However, this step only provides depth information to the dynamic regions that are overall consistent with other views. For a given video, some temporal high frequency content might not overlap with the image of the volume from that particular viewpoint. For example fire sparkles are video-specific and outside the volume. To assign depth to the remaining dynamic regions, we propagate the previously obtained foreground depth using the input video total variation as a guide. More specifically, we use a linear solver to solve a following least-squares system to propagate depth; details are provided in supplemental. Results of this depth propagation step can be seen in Figure 10.

The per-view geometry is approximate. Consequently, the depth map and its associated per-view mesh have high frequency noise producing visible artifacts when moving away from the input viewpoint. We smooth the mesh, reducing the maximum error by spreading inaccuracies over a larger region. This tradeoff is preferable because low-error regions remain plausible over a larger range of view angles. To reduce texture distortion during reprojection, we favor billboard-like geometry by smoothing the resulting depth maps in image space. We apply a bilateral filter on each depth map [TM98, PKT*09]. We set the spatial parameter that defines the radius of the image footprint of the filter to 50 pixels for full HD inputs, and the range parameter that controls the edge sensitivity to 10cm in world space.

Finally, we extract a per-view mesh from the depth map, by creating vertices for pixels corresponding to the dynamic effect.

5.4. Video matting

The dynamic regions in the scenes we target such as water and flames are often semi-transparent. In such cases, the color of a pixel in the dynamic region can be modeled as a mixture of the static background color and the dynamic object color using the compositing equation commonly used in natural image matting.

Instead, we use a modified version of the information-flow layer color estimation (IFL) method [AAP17, AOAP17], using the per-pixel temporal information to have a better background estimate. Details of the matting process can be found in supplemental. The final result of the matting process is shown in Figure 11. After this step we obtain a per-view background video and a per-view foreground video with alpha values. Our approach has similarities to Okabe et al. [OAO12] (Sec. 2.3). However, for each pixel they rely on optical flow to identify the most likely moments in the video where background is visible. In our case the dynamic elements have little low frequency motion, so for most dynamic pixels, the background is occluded during the entire video.

Figure 11: ***Final result of matting.*** *Left: final background result; Right: pre-multiplied foreground for a given frame.*

6. Rendering

We now have all the elements necessary for free-viewpoint navigation of scenes including stochastic phenomena such as flames, water streams, fountains etc., captured with a single video camera, such as a mobile phone.

At each frame, we choose the 4 candidate input views, and render the per-view meshes. We query and then reproject the RGBA textures from the foreground videos and perform per temporal frequency ULR blending independently. All ULR passes use per-pixel blending on the 3D projected position of the per-view meshes. All

the levels are summed to compose the final foreground layer. Finally we alpha-blend this foreground with a background ULR, using the alpha value extracted during matting. This background rendering reprojects background videos from 8 closest views onto the MVS geometry. As the background is more likely to be occluded than the foreground, more than 4 views are needed to cover the background.

In practice, there are two additional specific cases that frequently occur. The first involves dynamic phenomena (typically secondary illumination effects), that are projected onto well reconstructed geometry, e.g., the light cast by flames on the walls of a fireplace. The second involves the case where the dynamic phenomenon is fully transparent with two levels of depth that need to be treated separately, such as the surface of water and the sea floor.

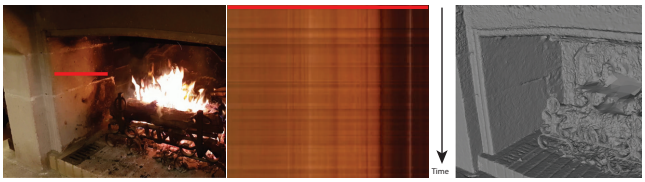

Figure 12: ***Time-dependent background.*** *The dynamic flames indirectly change the appearance of the static background in the input videos (left and horizontal patterns in the middle slice). However, textures corresponding to this background can be reprojected directly into novel views in a time-dependent manner using the reconstructed geometry (far right).*

To better reproduce the overall ambiance of the captured scene, these surfaces need to be rendered using time and view-dependent textures. To do this, we use the background videos computed from Sec. 5.4 as time-dependent textures, rendered with ULR weights. A standard occlusion-aware alpha compositing between the foreground and background layer is then performed (see Figure 12).

For fully transparent surfaces such as clear water, the different layers of the temporal Laplacian pyramid correspond to different visual phenomena. For water, the high frequencies are associated to wave displacement happening at the water surface while low frequencies are associated to the transmitted colors with waves canceled out. For such effects, we can extend the previous methods to not only have per-view geometries, but also per-frequency geometries. In the case of clear water, this allows distinct and correct parallax for both the waves and the sea floor. Note that this special case only works if, for any novel viewpoint, there is a visible low-frequency geometry behind the high-frequency geometry. Indeed, the temporal collapse of the renderings requires a low frequency residual, which is not guaranteed to exist in the general case if the geometry depends on the frequency.

In our case, the choice of rendering different frequency bands at different depths was determined manually per scene, by just specifying whether the effect is fully transparent or not.

7. Results and Evaluation

We implemented our method in C++, using OpenGL for the interactive rendering. Our method runs at 20 $\sim$ 30 fps on an NVidia GTX 1080, blending 4 videos with 7 frequency layers at each frame at a resolution of 1900x1440. Considering also the background videos, around 40 textures at 2K resolution are uploaded from RAM to VRAM at each frame, making the usage of the DXT5 compressed texture format essential for rendering performance.

Datasets, intermediate results data, and source code will be released `https://repo-sam.inria.fr/fungraph/vbr-dynamic-stationary/` (pending legal approval for the code).

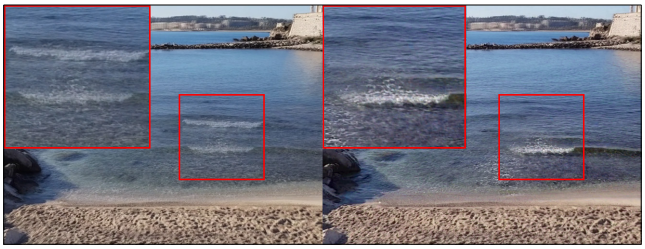

Figure 13: ***Results for the* Beach *scene.*** *Our per-frame improved version of ULR (left) [BBM*01] and our result (right). The per-frequency blending fields of our method are able to preserve high frequencies on the waves while ghosting is observed with standard ULR blending.*

Figure 14: ***Results for the* Seaside *scene.*** *Our per-frame improved version of ULR (left) [BBM*01] and our result (right). The low frequencies are reprojected onto the sea-floor while the high frequencies are reprojected onto the water plane, allowing correct parallax when moving the viewpoint.*

7.1. Results

All results are provided in the supplemental material, together with intermediate layers. We show a result for a fully opaque scene in

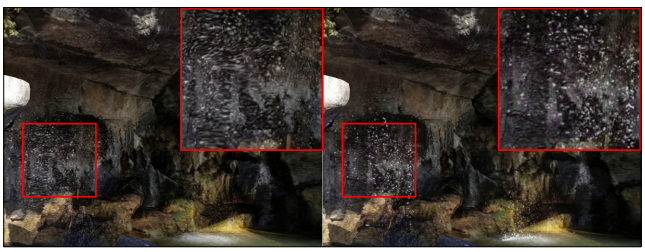

Figure 15: ***Results for the* Cave *scene.*** *Our per-frame improved version of ULR [BBM*01] (left), and our result (right).*

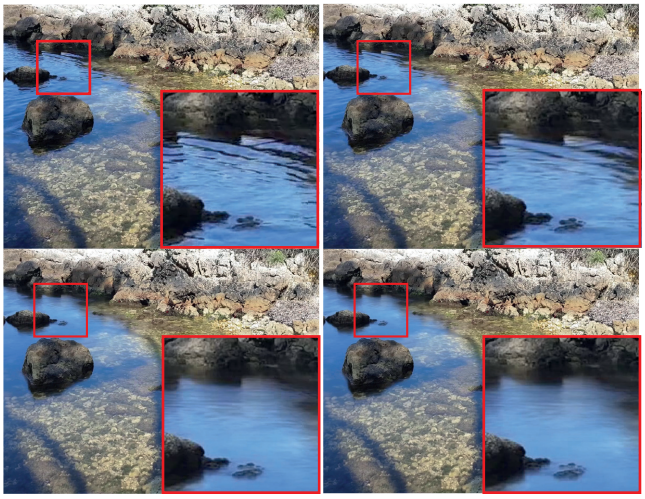

Figure 16: ***Scene editability*** *We can make use of the temporal frequencies decomposition to modify the scene appearance at run-time, by altering the contribution of each temporal frequency to the input videos. For example in the* Seaside *scene, it is possible to control the intensity of the ripples. Keeping the notation from Equation 1, we define the partially collapsed video as* $\mathcal{C}[V_i]_\ell = \sum_{k=\ell}^{n_L-1} \mathcal{K}[V_i]_k$. *We show frames of the videos* $\mathcal{C}[V_i]_\ell$, *from* $\ell = 0$ *(equal to the input video, top left) to* $\ell = 5$ *(only the three lowest frequencies are kept, bottom right). The relation* $\lambda\mathcal{C}[V_i]_\ell + (1-\lambda)\mathcal{C}[V_i]_{\ell+1} = \mathcal{C}[V_i]_\ell + \lambda\mathcal{K}[V_i]_{\ell+1}$, *for any* $0 \leq \lambda \leq 1$, *allows to linearly interpolate between two partial collapses at runtime, creating a continuous range for possible levels of dynamism in the scene. More examples are shown in the supplemental video.*

Figure 13. We can assume the sea is opaque in this scene since the parallax between the water surface and the sea-floor are similar when the sea-floor is visible. We also show a result for a fully transparent scene in Figure 14. For the *Seaside* scene, temporal frequencies are projected onto two different geometries, allowing correct parallax when moving the viewpoint. The highest frequencies are on the water plane while the low frequencies are able to cancel out the waves and allow us to see the sea-floor through the water. We show that by selecting a subset of the available frequencies, we have some control over the sea "liveliness" in Figure 16. Such editing can be done at runtime, by simply rendering and blending only a subset of the foreground pyramid layers.

We also provide results for two scenes where the effect is semi-transparent in Figure 17 and Figure 15. The *Fire* scene contains a fireplace which is highly time-dependent and semi-transparent. The fire also produces indirect lighting on the background chimney. The combination of both effects is able to produce the atmosphere of the input videos. The *Cave* scene contains two semi-transparent main waterfalls and an opaque water plane. There are also several indirect time-dependent lighting effects on the background rocks.

7.2. Evaluation

We are not aware of any other methods for free-viewpoint novel view synthesis from unsynchronized videos, which limits possible comparisons. We provide three baselines and an ablation study of the different components of our method. These baseline solutions are not designed to handle our data and thus are not expected to provide good quality results, but they provide an indication of the difficulty involved with processing our data.

We provide an extensive supplemental material page, showing different baselines and ablations side-by-side. We present qualitative comparisons to other methods. The unsynchronized video input makes it unclear how to define "ground truth" that could be used to allow either quantitative comparisons or meaningful perceptual tests with the small set of datasets we present. A user study could be envisaged with the generation of a large corpus of examples and an extensive large-population test, but this would be a research project on its own, and is beyond the scope of this project.

Baseline Comparisons. In Figure 18, we show the three baseline methods compared to our solution. First, we show per-frame reprojection on the basic multi-view reconstruction of the scene, built using both median videos and the still photos. We blend the input videos using Unstructured Lumigraph Rendering (ULR) weights [BBM*01].

The second baseline compares to a modified version of the Soft3D algorithm [PZ17]. Since Soft3D is designed for static environments, we create a scene from the median frame of each video and update the textures for each novel viewpoint, combining the rendered images into a video. The third baseline is inspired by [TKKT12, EDDM*08], but we use a state-of-the-art learning-based visibility-aware frame interpolation [JSJ*18]. Specifically, we first compute correspondences between two frames of two videos, then warp each frame by the homography over the view synthesis transition such that both are approximately aligned. We then compute frame interpolation using the Super-Slomo approach [JSJ*18], which directly generates the intermediate frame.

The results of the comparisons are best seen in the supplemental videos, since it is hard to appreciate the different tradeoffs between the methods in still images. The ULR solution clearly suffers from missing geometry (e.g., Fire and Cave scenes Figure 15) as well as having significant blur when blending (visible in the Beach scene Figure 13). The Soft3D baseline has difficulty dealing with the non photo-consistent nature of our input, and consequently suffers from severe temporal jittering and other artifacts due to the inherent sampling approach, based on a plane-sweep for each input video/photo. Finally, the Super-Slomo based interpolation provides a reasonable interpolation between close-by videos, but results in awkward warping artifacts compared to our geometry-guided solution.

Ablation Study. To illustrate the advantage brought by each step of our method, we present the following configurations: (1) Our blending using only MVS geometry and input videos instead of our localization; (2) Our dynamic element localization and our matting technique but using simple ULR blending; (3) Our method, but taking the closest depth in the voxel grid ray-cast instead of the median; (4) Discard the depth propagation step. (5) Our looping with ULR blending.

Not all ablations are possible for all scenes: For Beach and Seaside, the method with the proxy and our solution are the same and

thus are not included. Also, 3D localization and matting are not used for these scenes, so the corresponding ablations cannot be performed; a Table indicating which ablations are possible is included in supplemental for clarity.

The effect of each choice is clearly visible in the supplemental videos, and shown in Figure 17. When using global geometry we see obvious artifacts in scenes such as Fire and Cave, since the videos are incorrectly reprojected and are "smeared" onto the MVS geometry. Our matting and localization are able to reduce distortion and provide plausible parallax, but cannot completely remove the ghosting in the foreground. The effect of the voxel-grid raycast and the depth propagation are also clearly visible, since parts of the dynamic element are discarded. The study shows that each step of our method addresses different visual artifacts, and is indispensable in achieving the overall visual quality we provide.

7.3. Limitations

Our videos of real scenes do not represent perfectly stochastic dynamic textures. Some distinctive objects such as leaves on a water surface can actually be tracked over time, causing ghosting artifacts when appearing and disappearing during the loop. However, such artifacts are minor and the resulting loops remain visually plausible. We can handle motions that are small compared to the overall scene size, like branches swaying in the wind. Larger motions like a tree shaken by a violent storm would be challenging.

The per-view meshes we use are a first approximation, and do not provide perfect geometry for reprojection of the input videos. This can be seen in the transitions in the accompanying video, as blur artifacts and small distortions (for example in the fire). In addition, there is some slightly visible cross-fading, e.g., in the fire scene (Figure 17). This happens because the log and the flames slowly move as a whole over time. Our method cannot directly handle such non-stationary motion.

Scenes containing multiple effects of different nature are challenging for our method. The scene in Figure 19 has two stochastic dynamic phenomena, i.e., the fire and its heat induced distortion. While the fire can be represented as semi-transparent with matting, and the heat distortion as fully transparent, our method is unable to disambiguate between the two, leading to incorrect matting.

Finally, the initial volume carving approach imposes specific requirements on the capture conditions. If the dynamic effect we want to capture does not allow placement of video cameras in a way that volume carving will succeed, our method has difficulty representing the scene.

8. Conclusion

We have presented a first solution that, in the case of stochastic time-dependent effects, allows Video Based Rendering to be used while keeping similar simplicity of capture as Image Based Rendering, using a single camera. Our method provides both the casual capture and the free-viewpoint rendering of traditional wide-baseline IBR. We extended an image pyramid based decomposition to the time dimension to obtain video representations allowing seamless video looping and multi-view video blending for challenging scenes. Such a decomposition also allows some level of scene editing. By selecting which temporal frequency is used at runtime, the user can modify the degree of "liveliness" in the scene.

In conclusion, we proposed a method that provides a natural extension of IBR to dynamic scenes containing stationary stochastic phenomena, overcoming the time-dependency and unsynchronized video capture challenges.

For future work, we aim to better model the geometry of the effects and their possible volumetric properties, allowing better transitions between viewpoints. Indeed, the current median depth ap-

Figure 17: ***Ablation study for the* Fire *scene.*** *Top: Per-pixel ULR [BBM*01] (left) and our result (right). Our blending and matting reduces the ghosting artifacts when blending unsynchronized content. Middle: Our blending but using MVS geometry (left) and our matting with simple ULR blending (right). Our blending alone improves the baseline geometry IBR but still suffers from texture distortions and incorrect flame parallax. Matting alone reduces ghosting artifacts on the background but is not enough to handle ghosting on the foreground. Bottom: Our method without depth propagation step (left) and using the first encountered depth along the volume instead of the median (right). Without the depth propagation, video-specific dynamic parts might be discarded. By taking the first depth along the volume, depth is on average closer to what it should be. As a result, the geometry-based blending weights might discard some relevant part of the fire and incorrect parallax is more visible when moving the viewpoint.*

ULR [BBM*01]	Soft3D [PZ17]	Super-SloMo [JSJ*18]	Our Result

Figure 18: *Interpolation path between two viewpoints for the* Fire *scene. From left to right: Our per-frame improved version of ULR [BBM*01] (left) only using the two input viewpoints, Soft3D [PZ17] using all median videos for consensus pre-computation and all videos at runtime, the learning based Super-SloMo method [JSJ*18], used to generate an intermediate frame between two videos aligned using an homography, and our result using only the background and matting videos associated to the two input viewpoints.*

proach provides plausible consistency at the middle of the effect but the transition quality decreases near the per-view geometry boundaries. Moreover, the current visual hull approach is a limitation of the capture setup, forcing the user to have a wide angular coverage of the effect or manual intervention. Investigating more approximate methods for estimating geometry of surrounding effects and what would be the geometry accuracy needed for blending would be an interesting direction of future research.

Acknowledgments

This research was funded by the ERC Advanced grant FUNGRAPH No 788065 (http://fungraph.inria.fr), the doctoral fellowship of the Region Provence Alpes Cote d'Azur and the ANR project SEMAPOLIS (ANR-13-CORD-0003). The authors thank J. Tompkin for valuable feedback, S. Prakash for help with comparisons and the anonymous reviewers for their valuable feedback.

Appendix A: Laplacian Blending

Burt and Adelson described how to use image pyramids to blend two images seamlessly [BA83]. Their algorithm is based on the Gaussian pyramid $\mathcal{G}$ and the Laplacian pyramid $\mathcal{L}$ that are defined as follows for an image I and n_L levels:

$$\mathcal{G}[I]_0 = I \tag{5a}$$

$$\text{for } 0 < \ell < n_L, \quad \mathcal{G}[I]_\ell = \text{reduce}(G \otimes \mathcal{G}[I]_{\ell-1}) \tag{5b}$$

$$\text{for } 0 \leq \ell < n_L - 1, \quad \mathcal{L}[I]_\ell = \mathcal{G}[I]_\ell - \text{expand}(\mathcal{G}[I]_{\ell+1}) \tag{5c}$$

where G is a Gaussian kernel, $\otimes$ the convolution operator, and reduce$(\cdot)$ and expand$(\cdot)$ the operators that respectively halve and double the image resolution. Given two images I_1 and I_2, and a binary mask M that identifies a region of I_1 to be pasted into I_2, the algorithm builds the Laplacian pyramid of the output composite:

$$\mathcal{L}[O]_\ell \;=\; \mathcal{L}[I_1]_\ell \times \mathcal{G}[M]_\ell \;+\; \mathcal{L}[I_2]_\ell \times (1 - \mathcal{G}[M]_\ell) \tag{6}$$

where all the operations are performed per pixel. The final output O is reconstructed by collapsing its Laplacian pyramid, that is, repeatedly expanding and summing its levels.

In this work, we consider Laplacian pyramids on the temporal axis rather than spatial pyramids.

References

[AAP17] AKSOY Y., AYDIN T. O., POLLEFEYS M.: Information-flow matting, 2017. arXiv:1707.05055. 8

[AOAP17] AKSOY Y., OZAN AYDIN T., POLLEFEYS M.: Designing effective inter-pixel information flow for natural image matting. In *CVPR* (2017). 8

[AZP*05] AGARWALA A., ZHENG K. C., PAL C., AGRAWALA M., COHEN M., CURLESS B., SALESIN D., SZELISKI R.: Panoramic video textures. *ACM Trans. on Graphics (TOG) 24*, 3 (2005), 821–827. 2

[BA83] BURT P. J., ADELSON E. H.: A multiresolution spline with application to image mosaics. *ACM Trans. on Graphics 2*, 4 (1983), 217–236. 4, 5, 12

[BAAR12] BAI J., AGARWALA A., AGRAWALA M., RAMAMOORTHI R.: Selectively de-animating video. *ACM Trans. Graph. 31*, 4 (2012), 66–1. 2, 6

[BBM*01] BUEHLER C., BOSSE M., MCMILLAN L., GORTLER S., COHEN M.: Unstructured lumigraph rendering. In *Proceedings of the 28th annual conference on Computer graphics and interactive techniques* (2001), ACM, pp. 425–432. 1, 2, 4, 5, 9, 10, 11, 12

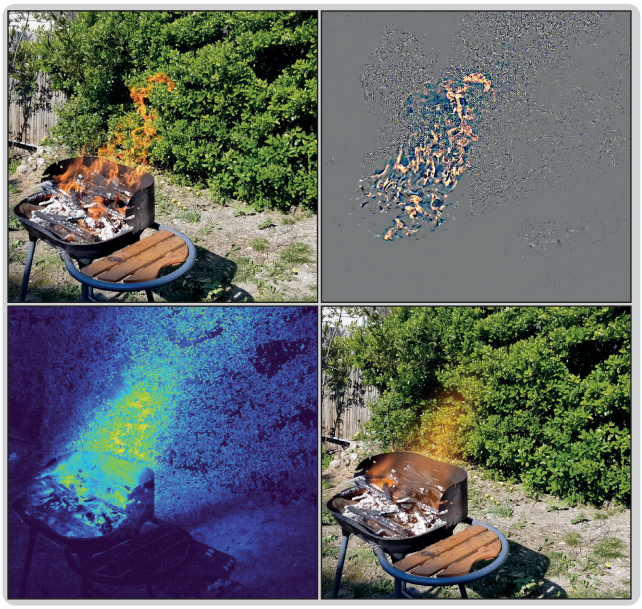

Figure 19: ***Failure case with multiple dynamic effects.*** *In this scene, the main dynamic effect is the fire (a frame from one input video, top left). However, the heat distortion induced by the fire is also an important source of temporal variation (per-pixel temporal total variation, bottom left). The heat distortion could be represented as a fully transparent effect, keeping all temporal frequencies but the residual (from the same input frame, highest temporal frequency at top right and residual at bottom right). However, our method is unable to disambiguate between the two effects: matting fails when fire is superposed over the vegetation covered by heat-distortion.*

[BBPP10] BALLAN L., BROSTOW G. J., PUWEIN J., POLLEFEYS M.: Unstructured video-based rendering: Interactive exploration of casually captured videos. In *ACM Trans. on Graphics (TOG)* (2010), vol. 29, ACM, p. 87. 2, 3

[BFO*20] BROXTON M., FLYNN J., OVERBECK R., ERICKSON D., HEDMAN P., DUVALL M., DOURGARIAN J., BUSCH J., WHALEN M., DEBEVEC P.: Immersive light field video with a layered mesh representation. *ACM Trans. on Graphics (TOG) 39*, 4 (2020), 86–1. 3

[CCS*15] COLLET A., CHUANG M., SWEENEY P., GILLETT D., EVSEEV D., CALABRESE D., HOPPE H., KIRK A., SULLIVAN S.: High-quality streamable free-viewpoint video. *ACM Trans. on Graphics (ToG) 34*, 4 (2015), 69. 2

[CDSHD13] CHAURASIA G., DUCHENE S., SORKINE-HORNUNG O., DRETTAKIS G.: Depth synthesis and local warps for plausible image-based navigation. *ACM Trans. on Graphics (TOG) 32*, 3 (2013), 30. 2

[CTMS03] CARRANZA J., THEOBALT C., MAGNOR M. A., SEIDEL H.-P.: Free-viewpoint video of human actors. *ACM transactions on graphics (TOG) 22*, 3 (2003), 569–577. 2

[EDDM*08] EISEMANN M., DE DECKER B., MAGNOR M., BEKAERT P., DE AGUIAR E., AHMED N., THEOBALT C., SELLENT A.: Floating textures. In *Computer graphics forum* (2008), vol. 27, pp. 409–418. 4, 10

[GKHH12] GREGSON J., KRIMERMAN M., HULLIN M. B., HEIDRICH W.: Stochastic tomography and its applications in 3d imaging of mixing fluids. *ACM Trans. Graph. 31*, 4 (2012), 52–1. 2, 3, 7

[HLSH17] HE M., LIAO J., SANDER P. V., HOPPE H.: Gigapixel panorama video loops. *ACM Trans. on Graphics (TOG) 37*, 1 (2017), 3. 3

[HPP*18] HEDMAN P., PHILIP J., PRICE T., FRAHM J.-M., DRETTAKIS G., BROSTOW G.: Deep blending for free-viewpoint image-based rendering. *ACM Trans. on Graphics 37*, 6 (2018). 1, 2, 4

[HRDB16] HEDMAN P., RITSCHEL T., DRETTAKIS G., BROSTOW G.: Scalable inside-out image-based rendering. *ACM Trans. on Graphics (TOG) 35*, 6 (2016), 231. 2

[JSJ*18] JIANG H., SUN D., JAMPANI V., YANG M.-H., LEARNED-MILLER E., KAUTZ J.: Super slomo: High quality estimation of multiple intermediate frames for video interpolation. In *CVPR* (2018). 10, 12

[Lau94] LAURENTINI A.: The visual hull concept for silhouette-based image understanding. *IEEE Trans. on pattern analysis and machine intelligence 16*, 2 (1994), 150–162. 7

[LGZL*20] LIU L., GU J., ZAW LIN K., CHUA T.-S., THEOBALT C.: Neural sparse voxel fields. *Advances in Neural Information Processing Systems 33* (2020). 2

[LJH13] LIAO Z., JOSHI N., HOPPE H.: Automated video looping with progressive dynamism. *ACM Trans. on Graphics (TOG) 32*, 4 (2013), 77. 3, 6

[LSS*19] LOMBARDI S., SIMON T., SARAGIH J., SCHWARTZ G., LEHRMANN A., SHEIKH Y.: Neural volumes: Learning dynamic renderable volumes from images. *ACM Trans. on Graphics* (2019). 2, 3

[LTK12] LEVIEUX P., TOMPKIN J., KAUTZ J.: Interactive viewpoint video textures. In *Proceedings of the 9th European Conference on Visual Media Production* (2012), ACM, pp. 11–17. 3

[MB20] MOJTABA BEMANA KAROL MYSZKOWSKI H.-P. S. T. R.: X-fields: Implicit neural view-, light- and time-image interpolation. *ACM Trans. on Graphics (Proc. SIGGRAPH Asia 2020) 39*, 6 (2020). doi:10.1145/3414685.3417827. 3

[MBR*00] MATUSIK W., BUEHLER C., RASKAR R., GORTLER S. J., MCMILLAN L.: Image-based visual hulls. In *Proceedings SIGGRAPH '00* (2000), pp. 369–374. 7

[MST*20] MILDENHALL B., SRINIVASAN P. P., TANCIK M., BARRON J. T., RAMAMOORTHI R., NG R.: Nerf: Representing scenes as neural radiance fields for view synthesis. In *ECCV* (2020). 2

[OAO12] OKABE M., ANJYO K., ONAI R.: Extracting fluid from a video for efficient post-production. In *Proceedings of the Digital Production Symposium* (2012), pp. 53–58. 3, 8

[ODAO15] OKABE M., DOBASHI Y., ANJYO K., ONAI R.: Fluid volume modeling from sparse multi-view images by appearance transfer. *ACM Trans. on Graphics (TOG) 34*, 4 (2015), 93. 3

[PCPMMN21] PUMAROLA A., CORONA E., PONS-MOLL G., MORENO-NOGUER F.: D-nerf: Neural radiance fields for dynamic scenes. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)* (jun 2021), IEEE. 2

[PGB03] PÉREZ P., GANGNET M., BLAKE A.: Poisson image editing. *ACM Trans. on graphics (TOG) 22*, 3 (2003), 313–318. 5

[PKT*09] PARIS S., KORNPROBST P., TUMBLIN J., DURAND F., ET AL.: Bilateral filtering: Theory and applications. *Foundations and Trends in Computer Graphics and Vision 4*, 1 (2009), 1–73. 8

[PZ17] PENNER E., ZHANG L.: Soft 3d reconstruction for view synthesis. *ACM Trans. on Graphics (TOG) 36*, 6 (2017), 235. 1, 2, 10, 12

[Rea18] REALITY C.: Realitycapture reconstruction software. https://www.capturingreality.com/Product, 2018. 2, 3

[RK20] RIEGLER G., KOLTUN V.: Free view synthesis. In *ECCV* (2020), Springer, pp. 623–640. 1, 2

[SF16] SCHÖNBERGER J. L., FRAHM J.-M.: Structure-from-motion revisited. In *CVPR* (2016). 2

[SH07] STARCK J., HILTON A.: Surface capture for performance-based animation. *IEEE computer graphics and applications 27*, 3 (2007), 21–31. 2

[SSSE00] SCHÖDL A., SZELISKI R., SALESIN D. H., ESSA I.: Video textures. In *Proceedings of the 27th annual conference on Computer graphics and interactive techniques* (2000), ACM Press/Addison-Wesley Publishing Co., pp. 489–498. 2

[SZPF16] SCHÖNBERGER J. L., ZHENG E., POLLEFEYS M., FRAHM J.-M.: Pixelwise view selection for unstructured multi-view stereo. In *ECCV* (2016). 2

[TBP10] TANEJA A., BALLAN L., POLLEFEYS M.: Modeling dynamic scenes recorded with freely moving cameras. In *Asian Conference on Computer Vision* (2010), Springer, pp. 613–626. 3

[TFT*20] TEWARI A., FRIED O., THIES J., SITZMANN V., LOMBARDI S., SUNKAVALLI K., MARTIN-BRUALLA R., SIMON T., SARAGIH J., NIESSNER M., PANDEY R., FANELLO S., WETZSTEIN G., ZHU J.-Y., THEOBALT C., AGRAWALA M., SHECHTMAN E., GOLDMAN D. B., ZOLLHÖFER M.: State of the Art on Neural Rendering. *Computer Graphics Forum (EG STAR 2020)* (2020). 2, 3

[TKKT12] TOMPKIN J., KIM K. I., KAUTZ J., THEOBALT C.: Videoscapes: exploring sparse, unstructured video collections. *ACM Trans. on Graphics (TOG) 31*, 4 (2012), 68. 2, 3, 10

[TLY20] THAPA S., LI N., YE J.: Dynamic fluid surface reconstruction using deep neural network. In *CVPR* (2020). 3

[TM98] TOMASI C., MANDUCHI R.: Bilateral filtering for gray and color images. In *ICCV* (1998), vol. 98. 8

[WBR07] WEINLAND D., BOYER E., RONFARD R.: Action recognition from arbitrary views using 3d exemplars. In *ICCV* (2007), IEEE. 2

[ZIT*18] ZANG G., IDOUCHI R., TAO R., LUBINEAU G., WONKA P., HEIDRICH W.: Space-time tomography for continuously deforming objects. *ACM Trans. on Graphics (TOG) 37*, 4 (2018), 100. 3

[ZIW*20] ZANG G., IDOUGHI R., WANG C., BENNETT A., DU J., SKEEN S., ROBERTS W. L., WONKA P., HEIDRICH W.: Tomofluid: Reconstructing dynamic fluid from sparse view videos. In *CVPR* (2020). 3

[ZRSK20] ZHANG K., RIEGLER G., SNAVELY N., KOLTUN V.: Nerf++: Analyzing and improving neural radiance fields. *arXiv preprint arXiv:2010.07492* (2020). 2