Eurographics Symposium on Rendering 2021
A. Bousseau and M. McGuire
(Guest Editors)

Volume 40 (2021), Number 4

Supplemental for Video-Based Rendering of Dynamic Stationary Environments from Unsynchronized Inputs

In this supplemental document we present technical details of various aspects of our method. We first present details on our seamless video looping approach, a detailed description of the video matting approach, details on the linear system used to propagate depth and finally some additional details on comparisons and ablations.

1. Seamless Video Looping

A naive option to create video loops is to restart the input sequence when it ends (Fig. 1a). This produces strong visual discontinuities 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 show that we can make our videos loop while being efficient, by using temporal Laplacian blending. As illustrated in Figure 1b, our approach is to have an overlap corresponding to half the duration of the input sequence. In this setup, there is always a transition happening between the first half and the second half of the sequence, and the output composite is a repetition of this transition. We generate this transition using multi-scale blending in the temporal domain, i.e., we apply Equation 1 and Equation 2, along the time axis instead of in the image plane, and use a mask that selects the first half of the sequence beginning, i.e., $M = 1$ in the first 25 % of the sequence. This resolves the question of the speed at which to cross-fade by adapting it to each temporal frequency band. The high frequencies transition sharply in the middle and the lower frequencies cross-fade over longer time spans.

$$\mathcal{G}[I]_0 = I \tag{1a}$$

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

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

$$\mathcal{L}[O]_\ell \;=\; \mathcal{L}[I_1]_\ell \times \mathcal{G}[M]_\ell \;+\; \mathcal{L}[I_2]_\ell \times (1 - \mathcal{G}[M]_\ell) \tag{2}$$

Discussion. Our adaptation of multi-scale blending to the time domain implicitly assumes that the observer's perception of temporal transitions is similar to their perception of the spatial transitions for which the original algorithm [BA83] was designed. Our experiments show that it is true to a first approximation and the discontinuities are greatly reduced (Figure 2). However, a slight temporal discontinuity can still be perceived. We hypothesize that this may be caused by the high frequencies at all the pixels transitioning at the same time, which creates a compound effect. We found that replacing the sharp mask associated to the lowest frequency by a smooth step over 10 frames produces better results at the cost of minimal ghosting that is imperceptible unless one pauses the video on one of these frames.

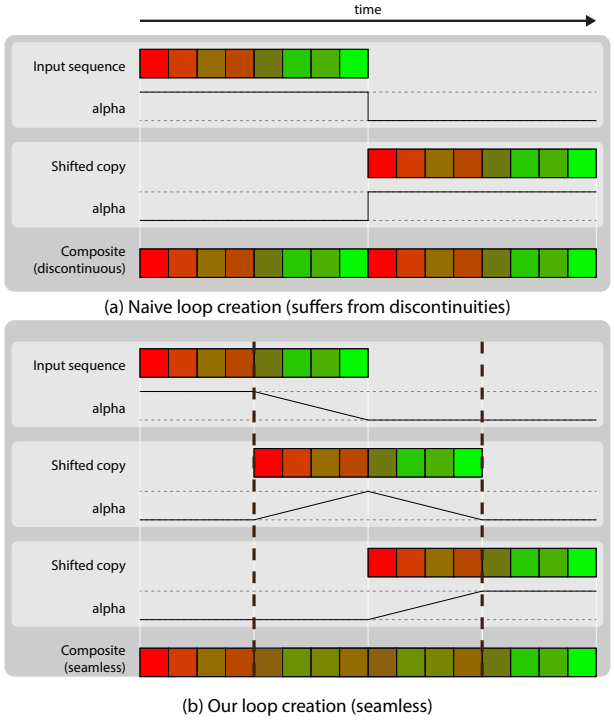

Figure 1: **Video looping using temporal blending.** Naive loop creation (top) simply plays the video back to back, which creates a temporal discontinuity at each loop end. Our loop creation (bottom), linearly blends temporally-shifted version of the input video, using a different blending profile for each temporal frequency band. The part delimited by the two dashed lines can by seamlessly played back to back.

Figure 2: **Video slices.** 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).

2. Video matting

The dynamic regions in the scenes we are interested in 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. If we consider the pixel p in a single frame from a single viewpoint, the compositing equation is written as:

$$\mathbf{c}_p = \alpha_p \mathbf{f}_p + (1-\alpha_p)\mathbf{b}_p, \tag{3}$$

where α_p is the opacity of the dynamic foreground, and $\mathbf{c}_p$, $\mathbf{f}_p$ and $\mathbf{b}_P$ are the observed pixel, dynamic foreground and static background colors in RGB, respectively. As α_p, $\mathbf{f}_p$ and $\mathbf{b}_P$ are all unknown in this equation, this is a highly under-constrained problem. This is typically solved using significant user intervention in the natural matting literature. Instead, we simplify the problem by taking advantage of our particular setup. One assumption we make is about $\mathbf{f}_p$. Since the foreground objects we consider are quite uniform in color, we use a single color $\mathbf{f}$ in the initial steps of our method. Also, given the static camera setup, we exploit the temporal information on the background to reliably determine $\mathbf{b}_p$ in a majority of the pixels. This way, we are able to get a clear opacity estimation with very little user input. We give the step-by-step explanation of our matting pipeline in the rest of this section.

Our matting pipeline requires a foreground color $\mathbf{f}$ as the only user input. This color is determined once for each dataset. For a given input frame (Fig. 4(a)), we use the binary mask (Fig. 4(b)) that marks the dynamic regions from the 3-D reconstruction as described in sections 5.2 and 5.3 in the paper, and proceed with opacity estimations for each viewpoint independently.

We first start by determining the fully foreground (i.e., opaque) and dynamic foreground regions in all the frames of a viewpoint using color-based segmentation. The dynamic foreground mask (Fig. 4(d)) represents the regions where the background is visible in some of the frames behind the dynamic objects. We reason about the dynamic and static (Fig. 4(c)) foreground regions by using the temporal median frame, i.e., the median color of each pixel in the temporal dimension (Fig. 4(e)), together with the input frame (Fig. 4(a)). Intuitively, a pixel is in the static foreground if its color is close to the median and close to the user-provided foreground color $\mathbf{f}$.

Figure 3: **Alpha-matte extraction:** (a) Initial temporally inpainted background. (b) Refined inpainted background. (c) Initial background with spatial inpainting. (d) Alpha matte α_p.

We mark the pixels that have a very similar color to $\mathbf{f}$ in both the median and the current frame as static foreground and reason that since α_p is likely unity in these regions, estimating $\mathbf{b}_p$ is of little importance.

The dynamic foreground regions, determined by good color matches to either the median or current frame, are used to determine regions (i.e., black pixels in Fig. 3(a)) for temporal inpainting of the background. This has the effect of adding a few additional background color pixels where available, Fig. 3(b).

In this setup, $\mathbf{b}_p$ is a determining factor for a reliable matte estimation. We therefore refine our temporal inpainting of the background in a second step. We first fill in the pixels where the background was never fully observed by a simple nearest-neighbor inpainting (Fig. 3(c)). We then use this intermediate background image for opacity estimation by solving (3) for α_p (Fig. 3(d)) We repeat the temporal inpainting of the background using this refined map by filling in $\mathbf{b}_p$ when $\alpha_p = 0$ in a frame, providing a small improvement to background estimation. This more complete background image is then ready for the final opacity estimation.

We determine the per-pixel per-frame opacities using this refined static background image. While we have modeled the foreground as a single color until now, for realistic rendering of the dynamic regions, we would like to determine the subtle color variations around the semi-transparent regions. For this purpose, we use a modified version of the information-flow layer color estimation (IFL) method [AAP17, AOAP17]. Layer color estimation methods typically aim to estimate $\mathbf{f}_p$ and $\mathbf{b}_p$ given the alpha matte and the original image. IFL defines a linear system of equations that includes energy definitions that target spatial consistency as well as nonlocal consistency between pixels based on color similarity for better stability in this under-constrained problem. The problem is better constrained in our case since we already have a

Figure 4: **Color-Based Segmentation:** (a) Input video frame and user-provided color **f**. (b) Binary mask for dynamic regions (c) Static foreground (opaque) (d) Dynamic foreground (e) Temporal median.

reliable estimation of the background colors. Hence, we replace the nonlocal energy terms in the IFL formulation with a quadratic cost, which encourages fidelity towards the estimated background colors. We keep the original no-transition flow term to encourage spatial smoothness in regions with small color and alpha gradients, and the original quadradic term that measures deviation from the compositing equation (Eq. (3)).

The final result of the matting process is shown in Figure 5. Our matting results are overall satisfactory. Further, since the matting does not use any background geometry information, it is sufficiently versatile to handle dynamic phenomena e.g., occluding the sky.

Figure 5: **Final result of matting.** Left: final background result; Right: pre-multiplied foreground for a given frame.

3. Depth propagation

This section describes the depth propagation step as part of the per-view geometries computation, as described in section 5.2 in the paper. We use a linear solver to solve the following least-squares system, with a data term E_d and a smoothness term E_s:

$$E(d_p) = E_d(p) + \sum_{q \in \mathcal{N}_p} E_s(p,q) \tag{4a}$$

$$E_d(p) = \begin{cases} \lambda_{\text{fg}} \left(d_p - d_p^{\text{fg}}\right)^2 \text{ if } d^{\text{fg}} \text{ is available} \\ \lambda_{\text{bg}} \left(d_p - d_p^{\text{bg}}\right)^2 \text{ otherwise} \end{cases} \tag{4b}$$

$$E_s(p,q) = \lambda_s (TV_p \cdot TV_q)^2 \tag{4c}$$

where d_p is the unknown per-pixel depth, d_p^{fg} is the foreground depth obtained from the voxel grid if available, d_p^{bg} is the background depth obtained from the static part of the MVS reconstruction, TV_p is the per-pixel RGB total variation L_1 norm, and $\mathcal{N}_p$ is the 4 neighborhood in image space, discarding neighbors across foreground occlusion edges. We use $\lambda_{\text{fg}} = 100, \lambda_{\text{bg}} = 1, \lambda_s = 0.01$ in all our experiments.

4. Comparisons

We present in Fig. 6, 5 frames replicating Fig. 18 in the main paper that shows 3 frames.

The full set of ablations is shown in Table 1. For the Beach and Seaside scenes the method with the proxy and our solution are the same and thus are not included; we label this as case 1 in the table. Also, 3D localization and matting are not used for these scenes, so the corresponding ablations cannot be performed, labeled as case 2. The scenarios are:

1. Our blending using only MVS geometry and input videos.
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 media.
4. Discard of the depth propagation step.
5. Looped ULR blending.

	Fire	Beach	Cave	Seaside
(1)	x	1	x	1
(2)	x	x	x	x
(3)	x	2	x	2
(4)	x	2	x	2
(5)	x	x	x	x

Table 1: Ablation study cases. All cases shown are marked with "x". For scenes with water there is no localization (case 1), so our method is the same as (1), and comparisons (3) and (4) do not apply (case 2).

References

[AAP17] AKSOY Y., AYDIN T. O., POLLEFEYS M.: Information-flow matting, 2017. arXiv:1707.05055. 2

[AOAP17] AKSOY Y., OZAN AYDIN T., POLLEFEYS M.: Designing effective inter-pixel information flow for natural image matting. In *CVPR* (2017). 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. 1

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

[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. 4

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

Figure 6: 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.

[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). 4

[LJH13] LIAO Z., JOSHI N., HOPPE H.: Automated video looping with progressive dynamism. *ACM Trans. on Graphics (TOG) 32*, 4 (2013), 77. 1

[PZ17] PENNER E., ZHANG L.: Soft 3d reconstruction for view synthesis. *ACM Trans. on Graphics (TOG) 36*, 6 (2017), 235. 4