Kernel-Based Frame Interpolation for Spatio-Temporally Adaptive Rendering

Karlis Martins Briedis
DisneyResearch|Studios
Zürich, Switzerland
ETH Zürich
Zürich, Switzerland
karlis.briedis@inf.ethz.ch

Abdelaziz Djelouah
DisneyResearch|Studios
Zürich, Switzerland
aziz.djelouah@disneyresearch.com

Raphaël Ortiz
DisneyResearch|Studios
Zürich, Switzerland
raphael.ortiz@disneyresearch.com

Mark Meyer
Pixar Animation Studios
Emeryville, California, USA
mmeyer@pixar.com

Markus Gross
DisneyResearch|Studios
Zürich, Switzerland
ETH Zürich
Zürich, Switzerland
grossm@inf.ethz.ch

Christopher Schroers
DisneyResearch|Studios
Zürich, Switzerland
christopher.schroers@disneyresearch.com

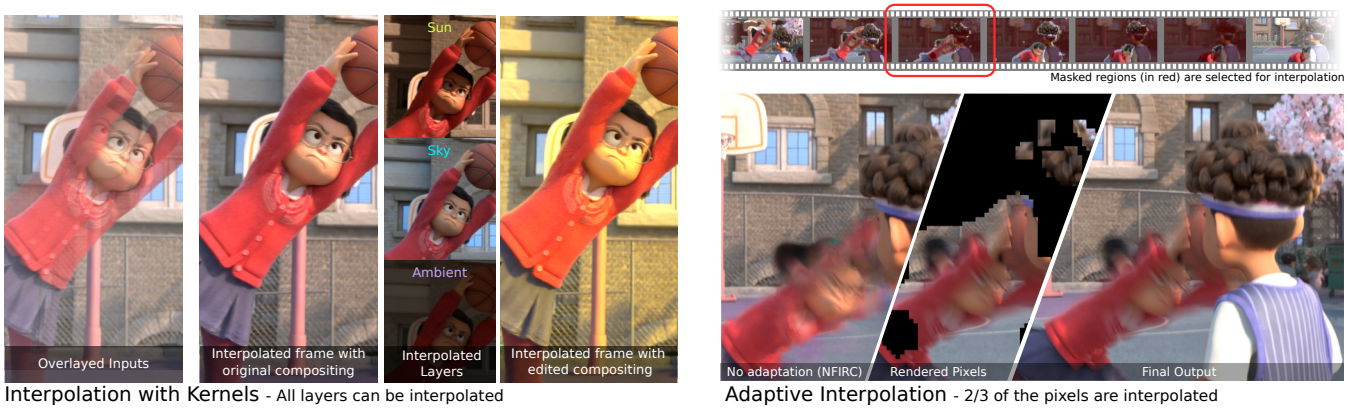

Figure 1: We propose a frame interpolation method for rendered content with two key features. First, a kernel based frame synthesis model which predicts the interpolated frame as a linear mapping of the input images. As a result, all the layers can be interpolated and a different composition can be created without any additional interpolation step. The second feature is an adaptive interpolation scheme where, for a given sequence of frames, decision on which image regions to render or interpolate is optimized: frame interpolation should be favored in image regions where it leads to good results, and rendering should be used in more challenging image patches. Images © 2023 Disney / Pixar

ABSTRACT

Recently, there has been exciting progress in frame interpolation for rendered content. In this offline rendering setting, additional inputs, such as albedo and depth, can be extracted from a scene at a very low cost and, when integrated in a suitable fashion, can significantly improve the quality of the interpolated frames. Although existing approaches have been able to show good results, most high-quality interpolation methods use a synthesis network for direct color prediction. In complex scenarios, this can result in unpredictable behavior and lead to color artifacts. To mitigate this and to increase robustness, we propose to estimate the interpolated frame by predicting spatially varying kernels that operate on image splats. Kernel prediction ensures a linear mapping from the input images to the output and enables new opportunities, such as consistent and efficient interpolation of alpha values or many other additional channels and render passes that might exist. Additionally, we present an adaptive strategy that allows predicting full or partial keyframes that should be rendered with color samples solely based on the auxiliary features of a shot. This content-based spatio-temporal adaptivity allows rendering significantly fewer color pixels as compared to a fixed-step scheme when wanting to maintain a certain quality. Overall, these contributions lead to a more robust method and significant further reductions of the rendering costs.

SIGGRAPH '23 Conference Proceedings, August 6–10, 2023, Los Angeles, CA, USA

ACM ISBN 979-8-4007-0159-7/23/08…$15.00
https://doi.org/10.1145/3588432.3591497

CCS CONCEPTS

• **Computing methodologies** → *Rendering*; *Reconstruction.*

KEYWORDS

Video Frame Interpolation, Rendered Content, Deep Learning

ACM Reference Format:
Karlis Martins Briedis, Abdelaziz Djelouah, Raphaël Ortiz, Mark Meyer, Markus Gross, and Christopher Schroers. 2023. Kernel-Based Frame Interpolation for Spatio-Temporally Adaptive Rendering. In *Special Interest Group on Computer Graphics and Interactive Techniques Conference Conference Proceedings (SIGGRAPH '23 Conference Proceedings), August 6–10, 2023, Los Angeles, CA, USA.* ACM, New York, NY, USA, 11 pages. https://doi.org/10.1145/3588432.3591497

1 INTRODUCTION

With the ever increasing demand for high quality rendered content, there has been a constant effort to reduce the very high costs associated with Monte Carlo renderings. To achieve that, commonly used methods include image denoising [Bako et al. 2017] or adaptive sampling [Kuznetsov et al. 2018] but these approaches do not leverage the redundancies across multiple frames. A more recent approach for re-using information across multiple frames is based on video frame interpolation [Briedis et al. 2021; Zimmer et al. 2015] or extrapolation [Guo et al. 2021], where the renderer produces a temporally downsampled sequence from which the missing frames are reconstructed. In this rendered setting, additional auxiliary feature buffers contain information about the scene and its motion, and they can be obtained at a low cost. It has been shown that methods utilizing them can achieve much better reconstruction than the purely image-based methods.

Despite recent progress, a few key challenges remain and the objective of this work is to address them. The first open challenge concerns the interpolation method itself. In a production context, frame interpolation cannot be limited to the final color. Other feature channels (such as alpha channel, decomposed per light contributions, etc.) also need to be interpolated, and the result must remain consistent between all of them in order to be used in subsequent processing such as compositing. Most of the existing video frame interpolation approaches employ a direct or residual prediction neural network for the final frame synthesis (including [Briedis et al. 2021]) which must be retrained for every channel combination while still not providing any guarantee for aligned outputs. Additionally, direct prediction networks with their unconstrained output range can produce color artifacts. The second important challenge, concerns adaptive interpolation. It is clear that on average the interpolation quality degrades with the number of skipped frames. However depending on the motion and occlusion in the scene, different interpolation intervals can be considered for each image region: larger for almost static background regions and smaller for fast moving objects in the scene. As a consequence simply interpolating entire frames is not optimal and better strategies are needed: frame interpolation should be favored in image regions where it leads to good results, and rendering should be favored on the rest.

The first part of our solution consists of a kernel-based frame interpolation model. In other words, the synthesis network relies on kernels, sharing coefficients across all the channels, to obtain the final output from the forward-warped keyframes. It ensures that the interpolation result is a linear combination of the inputs. Since the same kernels can be applied to an arbitrary number of layers, the interpolated frame has the same decomposition as the keyframes. As illustrated in Figure 1, it can therefore be edited and recomposed to create a different look. The second part is an implicit error prediction model, to estimate how good the interpolation network would perform for each pixel (or tile) given auxiliary features and a keyframe interval. This model is then leveraged in a spatiotemporally adaptive strategy to choose which pixels or tiles to render in a video as keyframe data. It significantly improves the interpolation quality compared to using a fixed sequence of full keyframes while rendering the same number of pixels (Figure 1).

To summarize, our contributions are:

- A new kernel-based frame interpolation model, that can be applied to an arbitrary number of channels (e.g. alpha) without adapting the method, while increasing robustness to color artifacts;
- An attention inspired mechanism, adapted to frame interpolation, with the ability to dynamically adjust kernel size for filling larger holes in warped keyframes.
- An adaptive interpolation strategy that includes an implicit interpolation error prediction model, to achieve better results for a given render budget (number of pixels or tiles);
- State-of-the-art results in frame interpolation for rendered content.

2 RELATED WORK

Different paradigms have been proposed for solving the problem of video frame interpolation. Phase-based methods [Meyer et al. 2018, 2015] model motion as a phase shift in the frequency domain, others use a feedforward neural network to predict the output frame directly, handling the motion implicitly [Choi et al. 2020; Kalluri et al. 2021; Long et al. 2016]. Our method is more closely related to methods with kernel estimation or motion-based image warping.

Similar to the direct prediction methods, prior kernel-based methods rely on implicit motion estimation by predicting spatially varying kernels that are applied to the source image. Niklaus *et al.* [2017a] propose estimating a dense local kernel for each of the output pixels, making it prohibitively expensive to use large kernels, thus greatly limiting the maximum motion they can handle. The size limitations still remain despite the estimation of separable convolution kernels [Niklaus et al. 2017b] and other improvements in the model and training [Niklaus et al. 2021]. AdaCoF [Lee et al. 2020] estimates spatially adaptive offsets and weights for deformable convolutions, allowing to reduce the kernel size, but large displacements are not well handled. Shi *et al.* [2022] extend this idea by using multi-frame processing at multiple scales. Ding *et al.* [2021] compress the weights of AdaCoF and extend it with a synthesis network, thus losing the desired properties of a kernel-based approach.

Building on the recent improvements in optical flow estimation [Hur and Roth 2019; Sun et al. 2018; Teed and Deng 2020], flow-based frame interpolation methods have typically performed best, thanks for example to the joint training of the optical flow

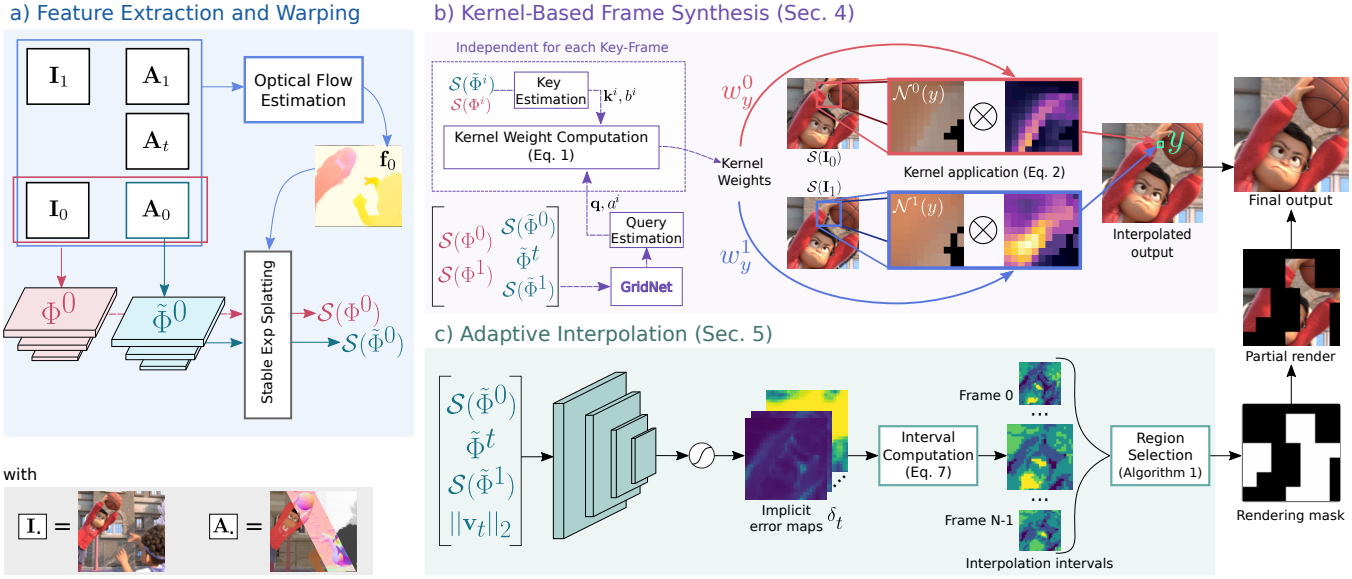

Figure 2: Method overview. Frame interpolation in CG context benefits from auxiliary buffers that are cheap to obtain for the frame to interpolate. a) We detail the process for one keyframe (I_0): we compute optical flow f_0 for splatting the context features Φ^i, $\tilde{\Phi}^i$ extracted by neural networks. b) Splatted features are used to estimate kernels for the final frame synthesis. See text (Section 4) for details c) Features extracted from auxiliary buffers are used to estimate implicit error maps, interpolation intervals, and regions to render. See text (Section 5) for details. Finally, the partial render is blended with the interpolated output for the final image. Images © 2023 Disney / Pixar

and frame interpolation [Jiang et al. 2018; Xue et al. 2019]. Multiple methods use backward warping with flows at the target time, obtained by warping keyframe flows [Bao et al. 2019, 2021; Lee et al. 2022], or estimating them directly [Danier et al. 2022; Lu et al. 2022; Park et al. 2020, 2021; Shangguan et al. 2022]. Others have explored forward warping with synthesis from forward-warped feature representations [Niklaus and Liu 2018], followed by differentiable splatting [Hu et al. 2022; Niklaus and Liu 2020]. To improve the motion estimation, [Chi et al. 2020; Liu et al. 2020; Xu et al. 2019] use higher order motion models estimated from an increased input context, while other methods focus on large motion [Reda et al. 2022; Sim et al. 2021] or efficient frame interpolation [Huang et al. 2022; Kong et al. 2022; Niklaus et al. 2023; Nottebaum et al. 2022].

Briedis *et al.* [2021] propose a method designed for rendered content, using auxiliary feature buffers to improve the results over prior methods. It follows an established set of image processing tasks for computer generated images: temporal processing [Zimmer et al. 2015], denoising [Bako et al. 2017; Vogels et al. 2018], and super-resolution [Xiao et al. 2020]. Guo *et al.* [2021] introduce a frame extrapolation solution that is tailored for real-time time applications, but does not benefit from the multiple keyframes available in the offline setting.

Our method aims to achieve the benefits of kernel-predicting approaches, while being able to support large motion that is typically only possible with flow-based methods. It is achieved by jointly combining and filtering out the artifacts in image splats. As such it is akin to guided image filtering tasks [Chen et al. 2016; Eisemann and Durand 2004; He et al. 2013; Kalantari et al. 2015; Li et al. 2012; Petschnigg et al. 2004; Tomasi and Manduchi 1998]. More recently, Işık *et al.* [2021] propose to use the cross-bilateral filter for Monte Carlo (MC) denoising with a predicted guidance map and a special care of the center pixel to allow suppression of bright outliers. Our dynamic weight prediction follows a similar idea, but estimates *asymmetric* guidance maps to performs frame merging and filtering of severe splatting artefacts in a joint step.

The most related works to our adaptive interpolation method are the estimation of error maps for error-aware frame interpolation ensembles [Chi et al. 2022], and the MC denoising for sampling distribution [Vogels et al. 2018]. Unlike the previous approaches, we estimate the error from just auxiliary feature buffers in an implicit formulation, and propose an algorithm for processing such maps to select regions to render for the beauty passes.

3 METHOD OVERVIEW

Given two consecutive frames $\mathbf{I}_0$ and $\mathbf{I}_1$, the objective of a frame interpolation method is to synthesize any intermediate frame $\mathbf{I}_t$ (with $t \in [0, 1]$). Contrary to live action videos, where only color information from the keyframes is available, frame interpolation in the rendered context can benefit from additional data $\mathbf{A}_i$ that is computationally cheap and easy to obtain (i.e. less compute than rendering the actual intermediate frame). As illustrated in Figure 2, we overall follow the same strategy used by existing frame interpolation works and in particular [Briedis et al. 2021]: for each keyframe $\mathbf{I}_i$ we compute optical flow $\mathbf{f}_i$ from $\mathbf{I}_i$ to $\mathbf{I}_t$ and a weight map for splatting the context features Φ^i, $\hat{\Phi}^i$ extracted by neural networks, before applying a compositing model.

For optical flow we use the approach proposed by [Briedis et al. 2021] that achieves non-linear motion estimation thanks to the

middle frame auxiliary buffers, and we refer to that work for more details. With the estimated motion, we obtain image splats $\mathcal{S}(\mathbf{I_i})$ from input image $\mathbf{I_i}$ and flow field $\mathbf{f_i}$ by using a numerically stable variant of softmax splatting with bilinear kernels similar to [Niklaus et al. 2023]. Details of the splatting are provided in the supplementary document.

Our first proposal is to use a kernel-predicting synthesis network to obtain the final output from the forward-warped inputs. This stems from the observation that splatting is a linear combination of the inputs, however existing direct synthesis models break this property. Our proposed solution ensures that the interpolation result is a linear combination of the inputs, through kernels sharing coefficients across all the channels.

This has several practical benefits over the existing approaches:

- a set of kernels can be estimated once and applied to an unlimited number of additional feature channels, e.g. alpha channel, with consistent results, and without a need to re-train the method for each of these channels;
- interpolation can be applied before or after image composition while maintaining the same outputs;
- as the output is limited to the convex hull of the input colors, the resulting images are well constrained and avoid color artifacts.

In the second part, we propose an adaptive interpolation strategy based on implicit error map estimation. We use a neural network to estimate how good the interpolation network would perform from just the auxiliary feature buffers at a given interval, compute the minimum interval to a rendered frame, and process the whole sequence to produce rendering masks.

As the region selection does not rely on the final images $\mathbf{I}_i$, the region selection and interpolation can be applied with just two rendering steps, where in the first pass only auxiliary buffers are rendered and in the second pass partial images are rendered based on the predicted rendering masks.

In the final processing step, interpolation outputs are blended with the partial renders.

4 KERNEL-BASED FRAME SYNTHESIS

We employ a GridNet [Fourure et al. 2017] that takes as input the motion compensated feature pyramids as in [Briedis et al. 2021], but instead of predicting the final 3-dimensional RGB colors, it outputs a 18-dimensional representation (see Figure 2b). In our design, it serves as query data in an attention-inspired mechanism [Vaswani et al. 2017]. As a result this predicted representation is expected to act as a reference to the missing frame and help with the kernel estimation.

Kernel Weight Prediction. We use an attention-inspired mechanism to predict per-pixel keys, queries, and scaling coefficients. These are used for a size-independent kernel estimation. The output of GridNet is decomposed into queries $\mathbf{q} \in \mathbb{R}^{H\times W\times D}$, where $D = 16$ is the per-pixel vector size, and $a^i \in \mathbb{R}^{H\times W\times 1}$ the scaling for each key-frame $i \in \{0, 1\}$. Keys $\mathbf{k}^i \in \mathbb{R}^{H\times W\times D}$ and biases b^i are estimated from the splatted feature pyramids Φ^i and $\tilde{\Phi}^i$ for each keyframe separately with two 1×1 convolution layers.

With the estimated coefficients and feature maps, we compute the weighting score w. For each output pixel $\boldsymbol{y}$, we estimate the relevance weight to any other pixel $\boldsymbol{x}$ in warp i with

$$w^i_{\boldsymbol{yx}} = -(a^i_{\boldsymbol{y}})^2 ||\mathbf{q}_{\boldsymbol{y}} - \mathbf{k}^i_{\boldsymbol{x}}||^2_2 + b^i_{\boldsymbol{x}}, \tag{1}$$

where we square the predicted scaling factor to constrain the weighting to be inversely proportional to the feature distance. The bias term b^i is motivated to allow down-weighting contributions from pixels in splats that can be viewed as outliers.

Kernel Application. Having the weighting metric, we can build a local kernel by limiting the distance between $\boldsymbol{y}$ and $\boldsymbol{x}$ to the neighborhood $\mathcal{N}$ around the center $\boldsymbol{y}$ (e.g. maximum of 5 pixel displacement for kernels of size 11×11). The output $\hat{\mathbf{I}}_t$ is synthesized as

$$\hat{\mathbf{I}}_t[\boldsymbol{y}] = \frac{\sum_i^N \sum_{\boldsymbol{x}\in\mathcal{N}(\boldsymbol{y})} \mathbf{M}^i_{\boldsymbol{x}} \exp(w^i_{\boldsymbol{yx}}) \mathcal{S}(\mathbf{I_i})[\boldsymbol{x}]}{\sum_i^N \sum_{\boldsymbol{x}\in\mathcal{N}(\boldsymbol{y})} \mathbf{M}^i_{\boldsymbol{x}} exp(w^i_{\boldsymbol{yx}}) + \varepsilon}, \tag{2}$$

where $N = 2$ is the number of input frames and $\mathbf{M}^i \in \{0, 1\}^{H\times W}$ a binary map that indicates holes in the warped frames. One major advantage of our approach is that $\mathcal{N}$ can be dynamically adjusted during inference to increase the perceptual window with bigger or dilated kernels, which is important to inpaint larger holes that would be challenging for directly predicted kernels with static size.

Dynamic-Offset Kernel Application. In practice, we only want to estimate kernels with large offsets in regions with holes in the splats. To this end, we propose using a dynamic per-pixel kernel size estimation and application. The offset γ is defined as the minimum offset such that at least Γ contributing pixels are present:

$$\gamma = \arg\min_{\gamma'} \quad \Gamma \leq \sum_{\boldsymbol{x}\in\eta(\boldsymbol{y},\gamma')} \mathbf{M}^i_{\boldsymbol{x}} \tag{3}$$

which can be used to define the pixel neighborhood

$$\mathcal{N}^i(\boldsymbol{y}) = \{\boldsymbol{x} \in \eta(\boldsymbol{y}, \gamma) | \quad \mathbf{M}^i_{\boldsymbol{x}} = 1\} \tag{4a}$$

$$\eta(\boldsymbol{y}', \gamma') = \{\boldsymbol{x}| \quad \|\boldsymbol{y}' - \boldsymbol{x}\|_\infty \leq \gamma'\} \tag{4b}$$

During training we use a constant offset $\gamma = 5$, but during inference the offset is dynamically adapted to ensure $\Gamma = 11^2$ contributing pixels.

Kernel Implementation. A naive implementation would compute Equation 2 and its partial derivatives w.r.t. inputs in a single kernel call. Instead, we propose to balance performance and operator flexibility/implementation complexity by disentangling per-frame weighted sum computations and relying on auto-differentiation. To achieve that, we re-write Equation 2, taking the factor $\mathbf{r_i}(\boldsymbol{y})$ outside of the inner sums as:

$$\mathbf{I}_t[\boldsymbol{y}] = \frac{\sum_i^N (\sum_{\boldsymbol{x}\in\mathcal{N}^i(\boldsymbol{y})} \hat{w}^i_{\boldsymbol{yx}} \mathcal{S}(\mathbf{I_i})[\boldsymbol{x}]) \mathbf{r_i}(\boldsymbol{y})}{\sum_i^N (\sum_{u,v\in\mathcal{N}(m,n)} \hat{w}^i_{\boldsymbol{yx}}) \mathbf{r_i}(\boldsymbol{y})} \tag{5a}$$

$$\hat{w}^i_{\boldsymbol{yx}} = \exp(w^i_{\boldsymbol{yx}} - \mathbf{m_i}(\boldsymbol{y})) \tag{5b}$$

$$\mathbf{r_i}(\boldsymbol{y}) = exp(\mathbf{m_i}(\boldsymbol{y}) - \max_{i\in 1..N} \mathbf{m_i}(\boldsymbol{y})) \tag{5c}$$

$$\mathbf{m_i}(\boldsymbol{y}) = \max_{\boldsymbol{x}\in\mathcal{N}^i(\boldsymbol{y})} w^i_{\boldsymbol{yx}} \tag{5d}$$

This allows us to compute and output three simpler quantities for each of the frames:

- $\sum_{\boldsymbol{x}\in\mathcal{N}^i(\boldsymbol{y})} \hat{w}^i_{\boldsymbol{yx}} \mathcal{S}(\mathbf{I_i})[\boldsymbol{x}]$
- $\sum_{\boldsymbol{x}\in\mathcal{N}(\boldsymbol{y})} \hat{w}^i_{\boldsymbol{yx}}$
- $\mathbf{m_i}(\boldsymbol{y})$

which are all numerically stable due to the subtraction of the largest weight exponentiation, and can be recombined by applying Eq. 5 on dense tensors to obtain the original values.

5 ADAPTIVE INTERPOLATION

A naive application of frame interpolation, where entire frames are skipped in favor of interpolation, already leads to a reduction in rendering time. On average the interpolation quality degrades with the number of skipped frames. However the error is often concentrated on relatively small and fast-moving objects, and a nearly static or slowly drifting background is correctly interpolated. Based on that observation, we propose a spatio-temporally adaptive interpolation: the gap between the keyframes is chosen dynamically per pixel or tile. In other words, a choice to render or not render is made per pixel or tile in the sequence, instead of full frames. This significantly improves the interpolation quality while rendering the same number of pixels.

We propose an approach with just two renderer invocations. In the first pass only the auxiliary feature buffers for each frame of the sequence are generated, as these are required at every frame and pixel. From the data of this first pass, we estimate a maximum keyframe gap for each of the tiles, i.e. *interpolation interval*, and generate rendering masks. In the second rendering pass we use the masks to render the necessary regions, then use frame interpolation for the remainder.

An overview of the method during inference is shown in Figure 2c.

5.1 Implicit Error Prediction

To choose an optimal interpolation interval, we first predict how good the interpolation network would perform from just the auxiliary feature buffers $\mathbf{A}_i$ as inputs (*albedo*, *normals*, *depth*, and *velocity*). For this task, we regress an implicit error map $\delta^m_t \in (0,1)$ at $\frac{1}{16}$ of the original resolution. Training is done by minimizing the following loss:

$$\mathcal{L}_{\text{Error}} = \mathcal{L}_{image}(\hat{\mathbf{I}}_t \cdot (1-\delta^m_t) + \mathbf{I}_t \cdot \delta^m_t, \mathbf{I}_t) + \lambda \cdot ||\delta^m_t||^2_2 \quad (6a)$$

$$\delta^m_t = \delta_t(\mathbf{A}_0, \mathbf{A}_t, \mathbf{A}_1) \quad (6b)$$

where $\hat{\mathbf{I}}_t$ is the interpolation result, $\mathbf{I}_t$ the ground truth frame at time t, and $\mathcal{L}_{image}$ is the chosen image loss. It is important to note again that the error prediction model δ_t only takes the auxiliary features as input.

The model δ_t shares many elements with the main method. Context features are extracted with and splatted, using the same optical flow model and warping technique. When computing the optical flow for the error network, we pass zeros for the color channels as they are not available. We show in section 6.4 that this produces a good approximation of the final motion. From the warped context features and the magnitude of velocity vectors $||\mathbf{v}_t||_2$, a VGG-style [Simonyan and Zisserman 2015] network with a sigmoid activation as a last layer is used to predict the error map.

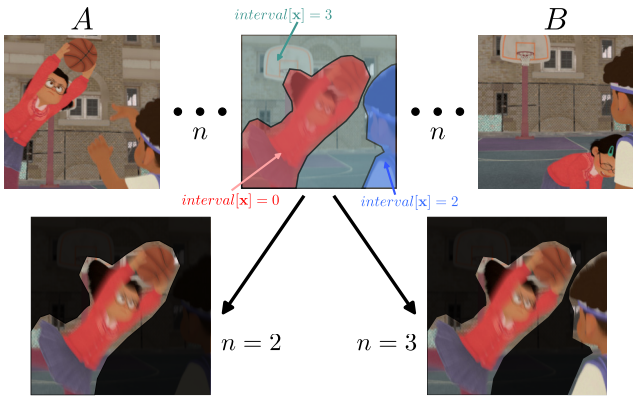

Figure 3: Illustration of the mask generation generation process. Given any sequence of frames, we consider the extreme ones (A and B) to be available. We need to decide which regions can be interpolated in the middle frame. The interpolation interval (see text for details) indicates the largest possible distance to a keyframe for every pixel. According to the keyframe distance n, different regions will be selected for rendering. The middle frame is then considered available, and becomes itself a keyframe for other intervals. Once all masks are computed, the beauty pass is done on all frames in parallel, followed by interpolation to fill the masked regions. Images © 2023 Disney / Pixar

The prediction is an *implicit* error map, because contrary to some other works [Vogels et al. 2018], we do not perform direct error regression (Equation. 6). A key advantage here is that error metrics typically depend on the values of the color, which are not available for the error-prediction network thus are hard to match. Additionally, it can be trained with the same image losses as the interpolation network, even when they do not provide per-pixel errors.

5.2 Interval Estimation and Mask Generation

We define the interpolation interval for each frame as the largest keyframe interval that allows interpolating the frame, while the error remains below a given threshold $p \in (0,1)$. More formally, the interval at a pixel/region $\boldsymbol{x}$ is defined as:

$$interval_t[\boldsymbol{x}] = \underset{k\in\mathbb{Z}\wedge 0\le k\le K}{\arg\max} \; \delta^k_t[\boldsymbol{x}] < p \quad (7a)$$

$$\text{with} \quad \delta^k_t = \delta_t(A_{t-k}, A_t, A_{t+k}) \quad \text{for} \quad k > 0 \quad (7b)$$

with $k = 0$ indicating no interpolation (i.e. always rendering) and hence $\delta^0_t = 0$. $K = 25$ is a chosen maximum one-sided interval. If one of the input frames does not exist (e.g. sequence beginning and end), we set $\delta^k_t = 1$.

In terms of control, the threshold p can be used as a knob by the artist to balance between interpolation quality and speed.

Region Selection. To generate masks, we start by marking the first and the last frame to be rendered and use them as the keyframes.

We then select the middle frame. For this middle frame, any pixel $\boldsymbol{x}$ that has an interpolation interval $interval_t[\boldsymbol{x}]$ smaller than the distance to the keyframe is selected for rendering (see Figure 3 for an illustration).

We then convert pixel-wise masks to tiles by expanding their boundaries with a Gaussian blur, and select every 64×64 tile with at least one marked pixel. After this, the middle frame is considered ready, and we process the resulting two sub-sequences in the same manner. A more formal description of the region selection is provided in Algorithm 1.

ALGORITHM 1: Rendering region selection

```
to_render ← {first_frame : full(), last_frame : full()}
q ← Queue{(first_frame, last_frame)})
while q not empty do
    lb, ub ← q.pop()
    mid ← round((lb + ub)/2)
    if lb + 1 < mid then q.push((lb, mid))
    if ub − 1 > mid then q.push((mid, ub))
    keyframe_distance ← max(mid − lb, ub − mid)
    render_mask = interval_mid < keyframe_distance
    render_mask = gaussian_blur(render_mask, σ = 1.0) ≥ 0.05
    render_mask = tiles_with_any_pixel(render_mask)
    if mean(render_mask) ≥ 95% then
        render_mask = full()
    end
    to_render[mid] = render_mask
end
return to_render
```

After having rendered the requested regions, we repeat the same order, interpolating non-rendered regions from $I_{\text{lb}}, I_{\text{ub}}$.

6 EXPERIMENTAL RESULTS

Implementation details. Our implementation follows [Briedis et al. 2021] where we replace the original splatting with our stable implementation, replace the shifted *ELU* [Clevert et al. 2016] weighting with softmax splatting [Niklaus and Liu 2020], and use our kernel-based frame synthesis approach. More details are provided in the supplementary document.

Handling linear RGB inputs. Preprocessing is necessary to handle linear RGB inputs that do not have a limited range. To support existing optical flow training procedures, usually in sRGB colorspace, we extend the linear→sRGB transform by applying a log transform for values > 1 (more details are in the supplementary). We use this transform for all inputs of the estimation networks, but, to maintain linearity, perform warping and apply the kernels in linear RGB. We observe that, by using this transform, flow networks handle HDR data well even if trained only with data in the range of $[0, 1]$.

Training. For the training schedule and data we follow [Briedis et al. 2021] and first train the networks with an L1 loss and then add the perceptual loss [Niklaus and Liu 2018] for the second stage with flow tuning. We compute the losses after applying the extended sRGB color transform to be able to use a pretrained VGG16 [Simonyan and Zisserman 2015] network.

Evaluation. We evaluate our method on 75 sequential frame triplets from movies that are not part of the training dataset. 25 of them are manually selected as challenging sequences, and the remaining 50 are randomly sampled. We reject trivial samples where the PSNR of overlaid inputs is larger than 40dB, or both [Bao et al. 2019] and [Briedis et al. 2021] achieve PSNR above 42dB, which we consider an almost perfect reconstruction. We report the average PSNR, SSIM, LPIPS[1] [Zhang et al. 2018], SMAPE [Vogels et al. 2018], and the median VMAF[2] value over the full dataset.

Runtime. It takes $0.76 \pm 0.01s$ to interpolate a single 1920×804px frame on an *NVIDIA RTX A6000* GPU. To interpolate 10 additional 3-channel layers along the main color, it takes $1.15 \pm 0.01s$. In the same setting, the baseline direct prediction scales linearly and takes $0.52 \pm 0.01s$ and $5.39 \pm 0.03s$ respectively.

6.1 Ablation Study

In our ablation study (Table 1) we first evaluate the importance of the different terms present in our kernel weight estimation model, namely: the scale $a^i_{\boldsymbol{y}}$, the bias $b^i_{\boldsymbol{x}}$ and the dot product compared to the L2 distance. Additionally, we compare against other synthesis approaches: direct color prediction (Direct Prediction), kernel prediction based on neural affinities (Affinity-based) as our adaptation of [Işık et al. 2021], direct Kernel prediction (KP), and a direct multi-scale KP network. Our proposed kernel based synthesis performs better than all the alternatives on all error measures. An outlier in the results is the Affinity-based approach, which suggests that *asymmetric* feature maps are necessary for the task of frame interpolation. We provide implementation details of these methods in the supplementary material.

Table 1: Ablative experiments of the proposed components. The first part evaluates choices made in our weight computation approach. The second part compares our method with alternative kernel-prediction (KP) variants and direct output synthesis (our baseline).

		PSNR ↑	SSIM ↑	LPIPS ↓	SMAPE ↓	VMAF ↑
Dynamic KP	w/o $a^i_{\boldsymbol{y}}$ (diverged)	n/a	n/a	n/a	n/a	n/a
	w/o $b^i_{\boldsymbol{x}}$	35.48	0.952	0.0548	3.193	89.36
	w/ dot product	34.97	0.950	0.0583	3.389	89.18
	Ours full	**35.73**	**0.953**	**0.0537**	**3.171**	**89.56**
Synthesis Method	Direct Prediction	35.65	0.952	0.0560	3.317	88.51
	Affinity-Based KP	32.73	0.936	0.0902	4.045	79.05
	Direct KP	34.96	0.949	0.0594	3.287	86.83
	Direct Multi-Scale KP	35.44	0.950	0.0572	3.213	86.72
	Ours full	**35.73**	**0.953**	**0.0537**	**3.171**	**89.56**

6.2 Comparison with Prior Methods

Comparisons against prior works are presented in Table 2. In this case adaptive interpolation is not used, and we just compare the core interpolation method on full frames. As our model takes advantage

[1] v0.1.2, 'alex' network
[2] https://github.com/Netflix/vmaf, v2.2.0

Table 2: Quantitative comparisons with prior methods. In the first block we report metrics for color-only methods. The second block contains methods designed for rendered content.

	PSNR ↑	SSIM ↑	LPIPS ↓	SMAPE ↓	VMAF ↑
DAIN [Bao et al. 2019]	27.81	0.882	0.1143	6.695	54.75
AdaCoF [Lee et al. 2020]	27.14	0.867	0.1427	7.604	41.44
BMBC [Park et al. 2020]	26.66	0.867	0.1454	7.736	45.88
CAIN [Choi et al. 2020]	27.24	0.868	0.1762	7.738	44.37
XVFI [Sim et al. 2021]	27.67	0.871	0.1349	7.061	48.94
XVFI (Vimeo)	26.88	0.868	0.1510	7.694	44.42
ABME [Park et al. 2021]	28.12	**0.889**	0.1426	6.912	48.82
VFIFormer [Lu et al. 2022]	27.99	0.886	0.1405	7.019	50.40
RIFE [Huang et al. 2022]	27.54	0.877	0.1137	6.994	51.65
FILM [Reda et al. 2022]	**28.66**	0.883	**0.1006**	**6.557**	**60.16**
NFIRC [Briedis et al. 2021]	35.62	0.952	0.0547	3.492	88.94
Ours	**35.73**	**0.953**	**0.0537**	**3.171**	**89.56**

of the additional information available for rendered content, it largely outperforms even most recent image-based video frame interpolation works [Lu et al. 2022; Reda et al. 2022].

Although our main objective was to increase the possibilities for frame interpolation through a kernel-based synthesis, our model also slightly outperforms the previous state-of-the-art interpolation method for rendered content from Briedis *et al.* [2021] (NFIRC). By relying on kernel application for image synthesis, our method is more robust and by construction cannot produce color artifacts that can sometimes be observed in the direct prediction outputs, as shown in Figure 6. Additional visual comparisons including color-only methods are shown in Figures 7-8.

6.3 Additional Channel Interpolation

The key motivation for our kernel-based interpolation is to allow the interpolation of any number of layers in a video sequence. This is already illustrated in Figure 1 for the decomposed per light contribution. In Figure 9 we show another possible application by interpolating the alpha channel. To obtain our results we apply the set of kernels estimated from the color RGB. For NFIRC [Briedis et al. 2021], as it does not support multi-channel interpolation, we run the network for a second time on the alpha channel (repeated across channel dimension, using the average of the outputs). To evaluate the consistency of both outputs, we perform alpha unpremultiplication, *i.e.* compute $color' = color/(alpha + 10^{-4})$, often done for color grading purposes. NFIRC shows significant artifacts due to inconsistent interpolation, while our method behaves similarly to the reference. In addition to the improved results, the possibility to apply the same estimated kernels also induces run-time savings, contrary to NFIRC [Briedis et al. 2021] that needs to be run for every selected layer.

6.4 Adaptive Interpolation

Training Details. To better handle velocity vectors instead of motion correspondences as produced by Pixar RenderMan, we double the training dataset size by adding shots from two additional

Table 3: Frame interpolation evaluation with modified flow network (FN) inputs.

FN Color inputs	PSNR	SSIM	LPIPS
RGB color	**35.73**	**0.953**	**0.0537**
Zeros	35.69	0.953	0.0541
Random Noise	35.65	0.953	0.0541

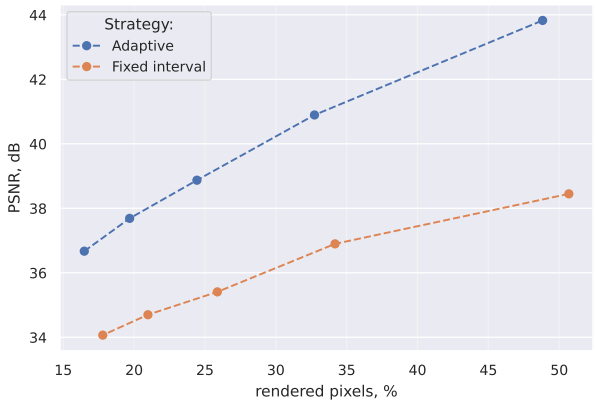

Figure 4: Frame interpolation with spatio-temporally adaptive vs. fixed intervals. Averaged over 7 shots from animation movies

movies. Training is done on sequential 3-frame sequences. During inference we increase the frame gap to the maximum of 25 frames. We use SMAPE as the image loss for training.

Re-using the Same Flow Network. Table 3 shows our method's evaluation with colors in the flow estimation network replaced by zeros or random noise. As only a small decrease in quality is observed, we conclude that the same network can be used to estimate motion just from the feature buffers without the color inputs.

Comparisons. We compare our adaptive interpolation with the baseline of equally-spaced full keyframes by skipping 2 to 6 frames. The adaptive method is run with the threshold value p that requires rendering fewer pixels. In our experiments, we use a 7-step binary search to find this threshold value. We evaluate the method on 7 shots with the total of 965 frames and report the average PSNR results in Figure 4. To avoid undefined PSNR values for rendered full keyframes, we perform full-shot measurements, *e.g.* for obtaining PSNR we first compute the mean squared error over the whole sequence. For the same (or lower) number of rendered pixels, the adaptive strategy significantly improves the quality of the results.

In Figure 11 we show the worse-case frame of both strategies for one of the sequences.

Runtime Breakdown. To evaluate the latency added by an additional auxiliary buffer rendering pass, we extend Blender's Cycles physically-based renderer to record per-tile rendering time and an option to render only feature buffers. We then use this to approximate the total latency of rendering on a 32-core CPU and *NVIDIA RTX A6000* GPU. It takes, on average, $0.30 \pm 0.01s$ to estimate the implicit error map, per single frame and interval, with 1920×804px frames, while the final mask selection is negligible,

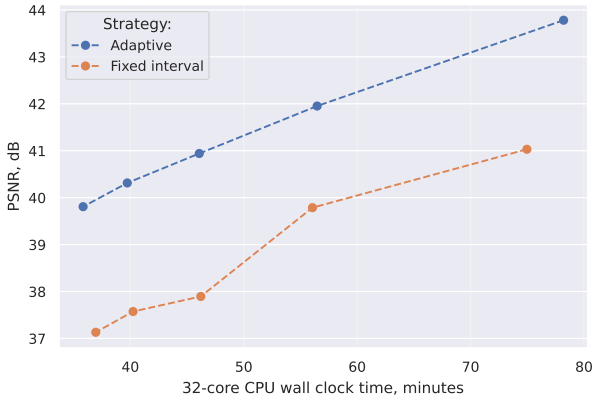

Figure 5: Frame interpolation with spatio-temporally adaptive vs. fixed intervals. Averaged over 3 Blender's shots

taking only 0.62*s* to process a whole 96-frame sequence. Figure 5 shows that, similarly to the proportion of rendered pixel, adopting an adaptive strategy is also well justified in terms of runtime. The overhead of rendering tiles from different frames is negligible. More details on the experimental setup and results are available in the supplementary material.

7 CONCLUSION

Our method is able to achieve production quality frame interpolation results by increasing robustness through an attention-inspired kernel prediction approach. It also significantly improves efficiency through introducing spatio-temporal adaptivity. As such we are able to take a step further into the wider adoption of frame interpolation as a standard tool for enabling cost savings in high quality rendering. There are still remaining challenges, in particular with interpolations that would require detail hallucination (see Figure 11), and promising avenues for future work. For example there is interesting potential for scenes with complex elements such as fluids, reflections and shadows by interpolating the different components separately. Regarding adaptive interpolation, focusing on capturing lighting changes and using all partial inputs could further increase efficiency. Another interesting area for further explorations is on how to optimally combine adaptive interpolation with the state-of-the-art temporal MC denoising methods.

ACKNOWLEDGMENTS

We thank Shilin Zhu and David Adler for their feedback and for providing the evaluation and training datasets. We also thank the anonymous reviewers for their helpful comments. The method was trained and tested on production imagery but the results were not part of the released productions.

REFERENCES

Steve Bako, Thijs Vogels, Brian McWilliams, Mark Meyer, Jan Novák, Alex Harvill, Pradeep Sen, Tony DeRose, and Fabrice Rousselle. 2017. Kernel-Predicting Convolutional Networks for Denoising Monte Carlo Renderings. *ACM Transactions on Graphics (Proceedings of SIGGRAPH 2017)* 36, 4, Article 97 (2017), 97:1–97:14 pages. https://doi.org/10.1145/3072959.3073708

Wenbo Bao, Wei-Sheng Lai, Chao Ma, Xiaoyun Zhang, Zhiyong Gao, and Ming-Hsuan Yang. 2019. Depth-Aware Video Frame Interpolation. In *IEEE Conference on Computer Vision and Pattern Recognition*.

Wenbo Bao, Wei-Sheng Lai, Xiaoyun Zhang, Zhiyong Gao, and Ming-Hsuan Yang. 2021. MEMC-Net: Motion Estimation and Motion Compensation Driven Neural Network for Video Interpolation and Enhancement. *IEEE Transactions on Pattern Analysis and Machine Intelligence* 43, 3 (2021), 933–948. https://doi.org/10.1109/TPAMI.2019.2941941

Karlis Martins Briedis, Abdelaziz Djelouah, Mark Meyer, Ian McGonigal, Markus Gross, and Christopher Schroers. 2021. Neural Frame Interpolation for Rendered Content. *ACM Trans. Graph.* 40, 6, Article 239 (dec 2021), 13 pages. https://doi.org/10.1145/3478513.3480553

Jiawen Chen, Andrew Adams, Neal Wadhwa, and Samuel W. Hasinoff. 2016. Bilateral Guided Upsampling. *ACM Trans. Graph.* 35, 6, Article 203 (dec 2016), 8 pages. https://doi.org/10.1145/2980179.2982423

Zhixiang Chi, Rasoul Mohammadi Nasiri, Zheng Liu, Juwei Lu, Jin Tang, and Konstantinos N. Plataniotis. 2020. All at Once: Temporally Adaptive Multi-frame Interpolation with Advanced Motion Modeling. In *Computer Vision – ECCV 2020*, Andrea Vedaldi, Horst Bischof, Thomas Brox, and Jan-Michael Frahm (Eds.). Springer International Publishing, Cham, 107–123.

Zhixiang Chi, Rasoul Mohammadi Nasiri, Zheng Liu, Yuanhao Yu, Juwei Lu, Jin Tang, and Konstantinos N Plataniotis. 2022. Error-Aware Spatial Ensembles for Video Frame Interpolation. *arXiv preprint arXiv:2207.12305* (2022).

Myungsub Choi, Heewon Kim, Bohyung Han, Ning Xu, and Kyoung Mu Lee. 2020. Channel Attention Is All You Need for Video Frame Interpolation. In *AAAI*.

Djork-Arné Clevert, Thomas Unterthiner, and Sepp Hochreiter. 2016. Fast and Accurate Deep Network Learning by Exponential Linear Units (ELUs). In *4th International Conference on Learning Representations, ICLR 2016, San Juan, Puerto Rico, May 2-4, 2016, Conference Track Proceedings*, Yoshua Bengio and Yann LeCun (Eds.). http://arxiv.org/abs/1511.07289

Duolikun Danier, Fan Zhang, and David Bull. 2022. ST-MFNet: A Spatio-Temporal Multi-Flow Network for Frame Interpolation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. 3521–3531.

Tianyu Ding, Luming Liang, Zhihui Zhu, and Ilya Zharkov. 2021. CDFI: Compression-Driven Network Design for Frame Interpolation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. 8001–8011.

Elmar Eisemann and Frédo Durand. 2004. Flash Photography Enhancement via Intrinsic Relighting. *ACM Trans. Graph.* 23, 3 (aug 2004), 673–678. https://doi.org/10.1145/1015706.1015778

D. Fourure, R. Emonet, E. Fromont, D. Muselet, A. Tremeau, and C. Wolf. 2017. Residual conv-deconv grid network for semantic segmentation. *arXiv preprint arXiv:1707.07958* (2017).

Jie Guo, Xihao Fu, Liqiang Lin, Hengjun Ma, Yanwen Guo, Shiqiu Liu, and Ling-Qi Yan. 2021. ExtraNet: Real-Time Extrapolated Rendering for Low-Latency Temporal Supersampling. *ACM Trans. Graph.* 40, 6, Article 278 (dec 2021), 16 pages. https://doi.org/10.1145/3478513.3480531

Kaiming He, Jian Sun, and Xiaoou Tang. 2013. Guided Image Filtering. *IEEE Transactions on Pattern Analysis and Machine Intelligence* 35, 6 (2013), 1397–1409. https://doi.org/10.1109/TPAMI.2012.213

Ping Hu, Simon Niklaus, Stan Sclaroff, and Kate Saenko. 2022. Many-to-many Splatting for Efficient Video Frame Interpolation. *2022 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)* (2022), 3543–3552.

Zhewei Huang, Tianyuan Zhang, Wen Heng, Boxin Shi, and Shuchang Zhou. 2022. Real-Time Intermediate Flow Estimation for Video Frame Interpolation. In *Proceedings of the European Conference on Computer Vision (ECCV)*.

Junhwa Hur and Stefan Roth. 2019. Iterative Residual Refinement for Joint Optical Flow and Occlusion Estimation. In *CVPR*.

Mustafa Işık, Krishna Mullia, Matthew Fisher, Jonathan Eisenmann, and Michaël Gharbi. 2021. Interactive Monte Carlo Denoising Using Affinity of Neural Features. *ACM Trans. Graph.* 40, 4, Article 37 (jul 2021), 13 pages. https://doi.org/10.1145/3450626.3459793

Huaizu Jiang, Deqing Sun, Varun Jampani, Ming-Hsuan Yang, Erik Learned-Miller, and Jan Kautz. 2018. Super slomo: High quality estimation of multiple intermediate frames for video interpolation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 9000–9008.

Nima Khademi Kalantari, Steve Bako, and Pradeep Sen. 2015. A Machine Learning Approach for Filtering Monte Carlo Noise. *ACM Trans. Graph.* 34, 4, Article 122 (jul 2015), 12 pages. https://doi.org/10.1145/2766977

Tarun Kalluri, Deepak Pathak, Manmohan Chandraker, and Du Tran. 2021. FLAVR: Flow-Agnostic Video Representations for Fast Frame Interpolation. (2021).

Lingtong Kong, Boyuan Jiang, Donghao Luo, Wenqing Chu, Xiaoming Huang, Ying Tai, Chengjie Wang, and Jie Yang. 2022. IFRNet: Intermediate Feature Refine Network for Efficient Frame Interpolation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. 1969–1978.

Alexandr Kuznetsov, Nima Khademi Kalantari, and Ravi Ramamoorthi. 2018. Deep adaptive sampling for low sample count rendering. In *Computer Graphics Forum*, Vol. 37. Wiley Online Library, 35–44.

Hyeongmin Lee, Taeoh Kim, Tae young Chung, Daehyun Pak, Yuseok Ban, and Sangyoun Lee. 2020. AdaCoF: Adaptive Collaboration of Flows for Video Frame Interpolation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*.

Sungho Lee, Narae Choi, and Woong Il Choi. 2022. Enhanced Correlation Matching based Video Frame Interpolation. In *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision*. 2839–2847.

Tzu-Mao Li, Yu-Ting Wu, and Yung-Yu Chuang. 2012. SURE-Based Optimization for Adaptive Sampling and Reconstruction. *ACM Trans. Graph.* 31, 6, Article 194 (nov 2012), 9 pages. https://doi.org/10.1145/2366145.2366213

Yihao Liu, Liangbin Xie, Li Siyao, Wenxiu Sun, Yu Qiao, and Chao Dong. 2020. Enhanced quadratic video interpolation. In *European Conference on Computer Vision Workshops*.

Gucan Long, Laurent Kneip, Jose M Alvarez, Hongdong Li, Xiaohu Zhang, and Qifeng Yu. 2016. Learning image matching by simply watching video. In *European Conference on Computer Vision*. Springer, 434–450.

Liying Lu, Ruizheng Wu, Huaijia Lin, Jiangbo Lu, and Jiaya Jia. 2022. Video Frame Interpolation With Transformer. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. 3532–3542.

Simone Meyer, Abdelaziz Djelouah, Brian McWilliams, Alexander Sorkine-Hornung, Markus Gross, and Christopher Schroers. 2018. PhaseNet for Video Frame Interpolation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*.

Simone Meyer, Oliver Wang, Henning Zimmer, Max Grosse, and Alexander Sorkine-Hornung. 2015. Phase-Based Frame Interpolation for Video. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 1410–1418. https://doi.org/10.1109/CVPR.2015.7298747

Simon Niklaus, Ping Hu, and Jiawen Chen. 2023. Splatting-Based Synthesis for Video Frame Interpolation. In *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision (WACV)*. 713–723.

Simon Niklaus and Feng Liu. 2018. Context-Aware Synthesis for Video Frame Interpolation. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*.

Simon Niklaus and Feng Liu. 2020. Softmax Splatting for Video Frame Interpolation. In *IEEE Conference on Computer Vision and Pattern Recognition*.

Simon Niklaus, Long Mai, and Feng Liu. 2017a. Video Frame Interpolation via Adaptive Convolution. In *IEEE Conference on Computer Vision and Pattern Recognition*.

Simon Niklaus, Long Mai, and Feng Liu. 2017b. Video Frame Interpolation via Adaptive Separable Convolution. In *IEEE International Conference on Computer Vision*.

Simon Niklaus, Long Mai, and Oliver Wang. 2021. Revisiting Adaptive Convolutions for Video Frame Interpolation. In *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision (WACV)*. 1099–1109.

Moritz Nottebaum, Stefan Roth, and Simone Schaub-Meyer. 2022. Efficient Feature Extraction for High-resolution Video Frame Interpolation. In *British Machine Vision Conference BMVC*.

Junheum Park, Keunsoo Ko, Chul Lee, and Chang-Su Kim. 2020. BMBC: Bilateral Motion Estimation with Bilateral Cost Volume for Video Interpolation. In *Computer Vision – ECCV 2020*, Andrea Vedaldi, Horst Bischof, Thomas Brox, and Jan-Michael Frahm (Eds.). Springer International Publishing, Cham, 109–125.

Junheum Park, Chul Lee, and Chang-Su Kim. 2021. Asymmetric Bilateral Motion Estimation for Video Frame Interpolation. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*. 14539–14548.

Georg Petschnigg, Richard Szeliski, Maneesh Agrawala, Michael Cohen, Hugues Hoppe, and Kentaro Toyama. 2004. Digital Photography with Flash and No-Flash Image Pairs. *ACM Trans. Graph.* 23, 3 (aug 2004), 664–672. https://doi.org/10.1145/1015706.1015777

Fitsum Reda, Janne Kontkanen, Eric Tabellion, Deqing Sun, Caroline Pantofaru, and Brian Curless. 2022. FILM: Frame Interpolation for Large Motion. In *European Conference on Computer Vision (ECCV)*.

Wentao Shangguan, Yu Sun, Weijie Gan, and Ulugbek S. Kamilov. 2022. Learning Cross-Video Neural Representations for High-Quality Frame Interpolation. In *Proc. European Conference on Computer Vision (ECCV)*. Tel Aviv, Israel, 511–528.

Zhihao Shi, Xiangyu Xu, Xiaohong Liu, Jun Chen, and Ming-Hsuan Yang. 2022. Video Frame Interpolation Transformer. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. 17482–17491.

Hyeonjun Sim, Jihyong Oh, and Munchurl Kim. 2021. XVFI: eXtreme Video Frame Interpolation. In *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*.

Karen Simonyan and Andrew Zisserman. 2015. Very Deep Convolutional Networks for Large-Scale Image Recognition. In *International Conference on Learning Representations*.

Deqing Sun, Xiaodong Yang, Ming-Yu Liu, and Jan Kautz. 2018. Pwc-net: Cnns for optical flow using pyramid, warping, and cost volume. In *Proceedings of the IEEE conference on computer vision and pattern recognition*. 8934–8943.

Zachary Teed and Jia Deng. 2020. RAFT: Recurrent All-Pairs Field Transforms for Optical Flow. In *Computer Vision - ECCV 2020 - 16th European Conference (Lecture Notes in Computer Science, Vol. 12347)*. Springer, 402–419. https://doi.org/10.1007/978-3-030-58536-5_24

C. Tomasi and R. Manduchi. 1998. Bilateral filtering for gray and color images. In *Sixth International Conference on Computer Vision (IEEE Cat. No.98CH36271)*. 839–846. https://doi.org/10.1109/ICCV.1998.710815

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Ł ukasz Kaiser, and Illia Polosukhin. 2017. Attention is All you Need. In *Advances in Neural Information Processing Systems*, I. Guyon, U. Von Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett (Eds.), Vol. 30. Curran Associates, Inc. https://proceedings.neurips.cc/paper/2017/file/3f5ee243547dee91fbd053c1c4a845aa-Paper.pdf

Thijs Vogels, Fabrice Rousselle, Brian McWilliams, Gerhard Röthlin, Alex Harvill, David Adler, Mark Meyer, and Jan Novák. 2018. Denoising with Kernel Prediction and Asymmetric Loss Functions. *ACM Transactions on Graphics (Proceedings of SIGGRAPH 2018)* 37, 4, Article 124 (2018), 124:1–124:15 pages. https://doi.org/10.1145/3197517.3201388

Lei Xiao, Salah Nouri, Matt Chapman, Alexander Fix, Douglas Lanman, and Anton Kaplanyan. 2020. Neural Supersampling for Real-Time Rendering. *ACM Trans. Graph.* 39, 4, Article 142 (July 2020), 12 pages. https://doi.org/10.1145/3386569.3392376

Xiangyu Xu, Li Siyao, Wenxiu Sun, Qian Yin, and Ming-Hsuan Yang. 2019. Quadratic Video Interpolation. In *Advances in Neural Information Processing Systems*, H. Wallach, H. Larochelle, A. Beygelzimer, F. dÀlché-Buc, E. Fox, and R. Garnett (Eds.), Vol. 32. Curran Associates, Inc. https://proceedings.neurips.cc/paper/2019/file/d045c59a90d7587d8d671b5f5aec4e7c-Paper.pdf

Tianfan Xue, Baian Chen, Jiajun Wu, Donglai Wei, and William T Freeman. 2019. Video enhancement with task-oriented flow. *International Journal of Computer Vision* 127, 8 (2019), 1106–1125.

Richard Zhang, Phillip Isola, Alexei A Efros, Eli Shechtman, and Oliver Wang. 2018. The Unreasonable Effectiveness of Deep Features as a Perceptual Metric. In *CVPR*.

Henning Zimmer, Fabrice Rousselle, Wenzel Jakob, Oliver Wang, David Adler, Wojciech Jarosz, Olga Sorkine-Hornung, and Alexander Sorkine-Hornung. 2015. Path-space Motion Estimation and Decomposition for Robust Animation Filtering. *Computer Graphics Forum (Proceedings of EGSR)* 34, 4 (June 2015). https://doi.org/10/f7mb34

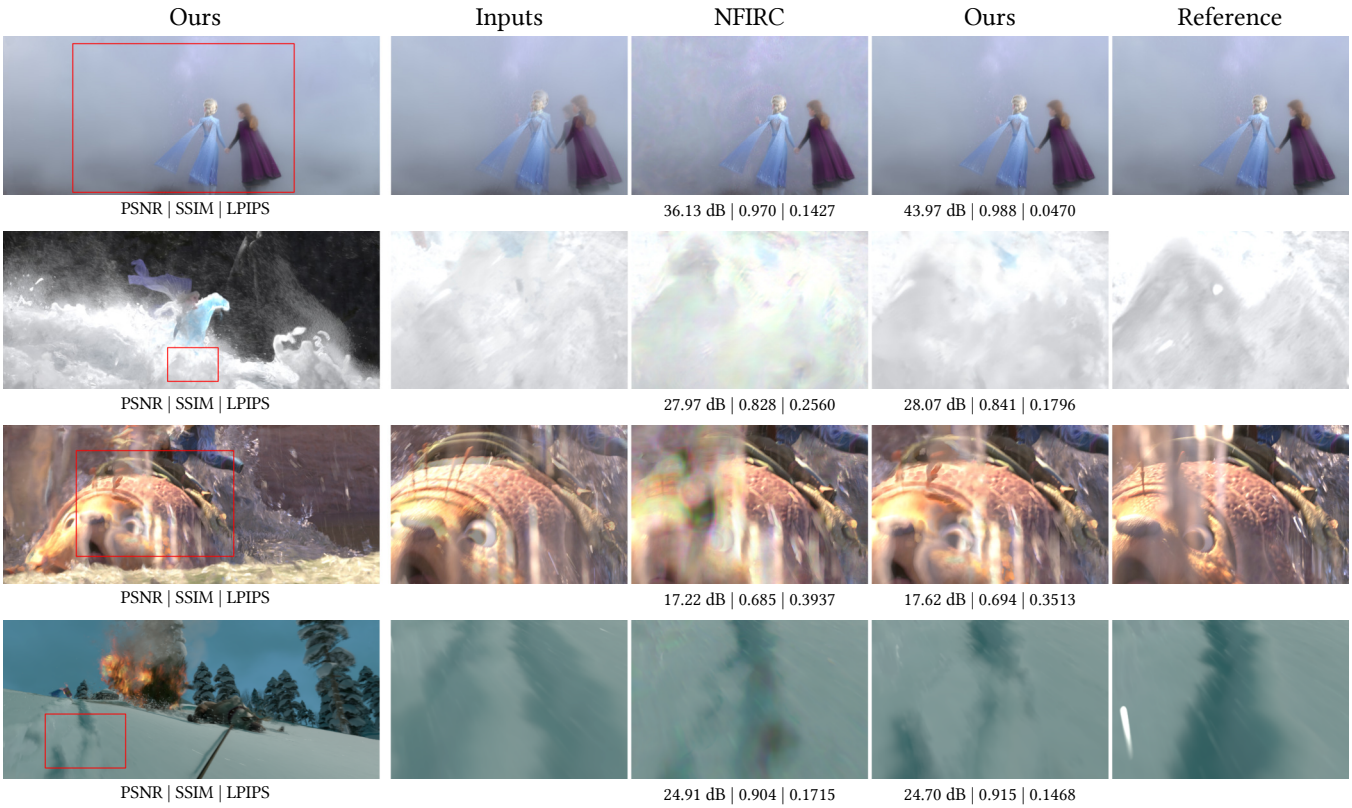

Figure 6: Qualitative comparison with the NFIRC [Briedis et al. 2021] method. It can be observed that our method is much more robust to color artifacts than the prior direct prediction method. © 2023 Disney

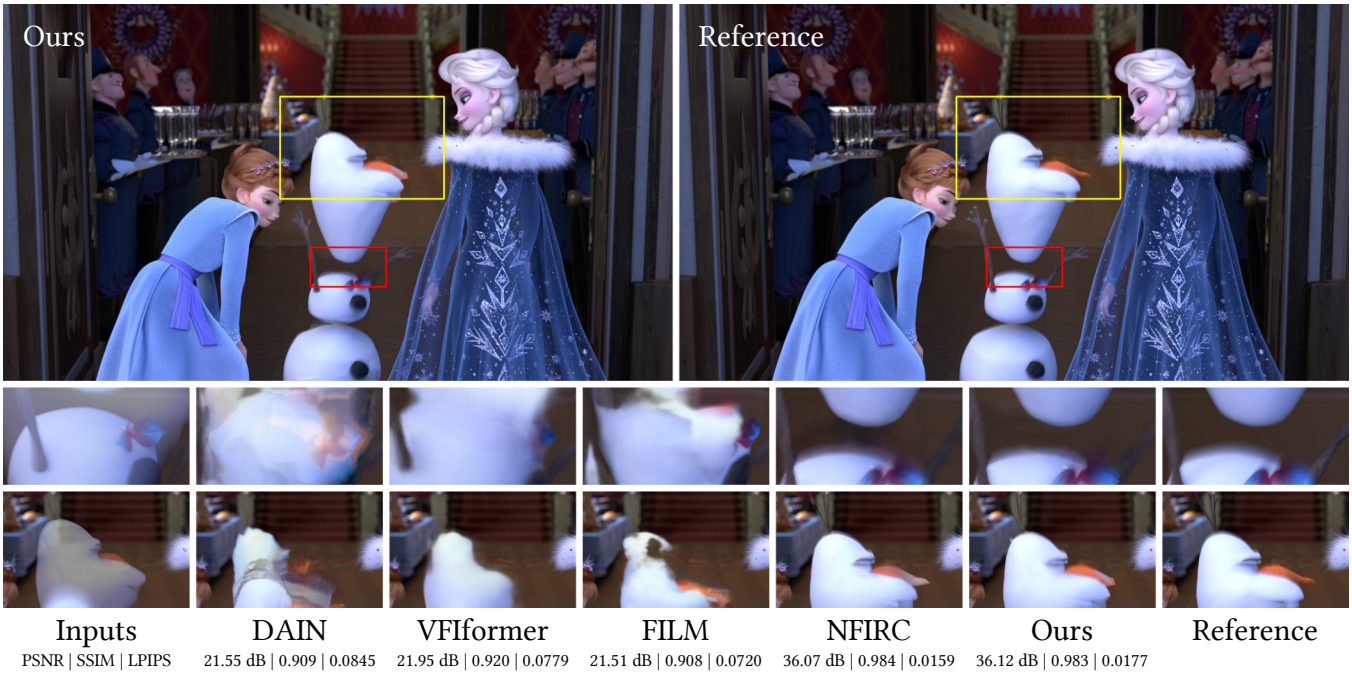

Figure 7: Visual comparison with both color-only and renderings frame interpolation methods. © 2023 Disney

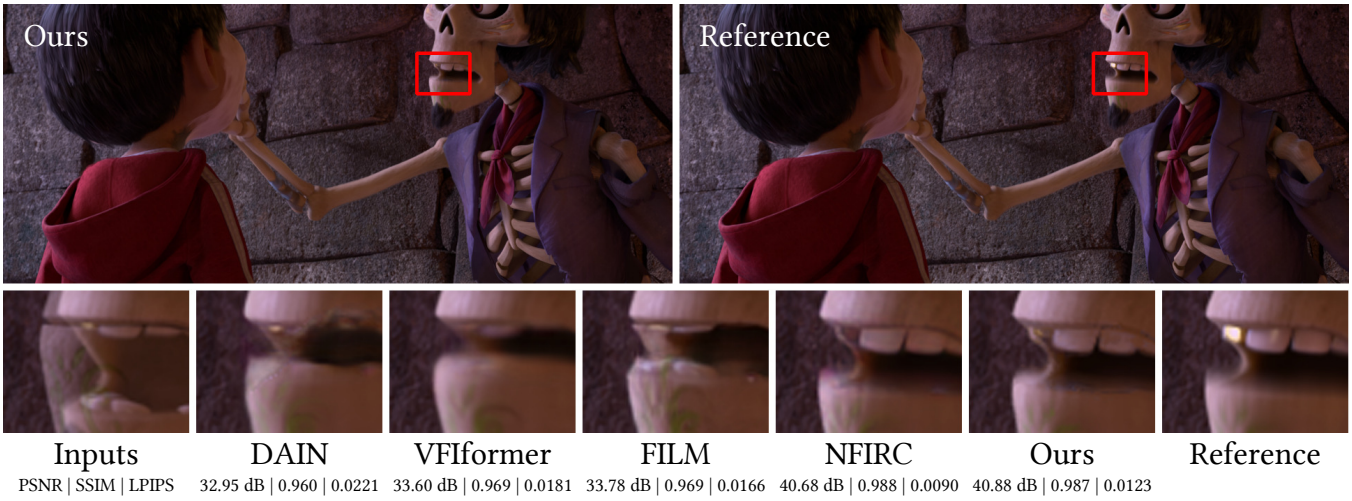

Figure 8: Visual comparison with both color-only and renderings frame interpolation methods. © 2023 Disney / Pixar

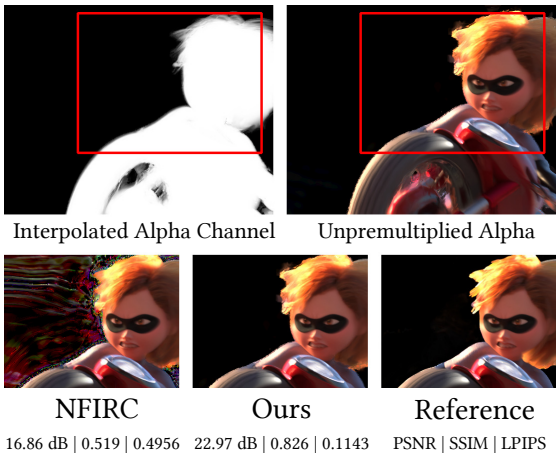

Figure 9: Results of the alpha channel interpolation, visualized by performing alpha unpremultiplication with the interpolated color and alpha. See the text (Section 6.3) for more details. © 2023 Disney / Pixar

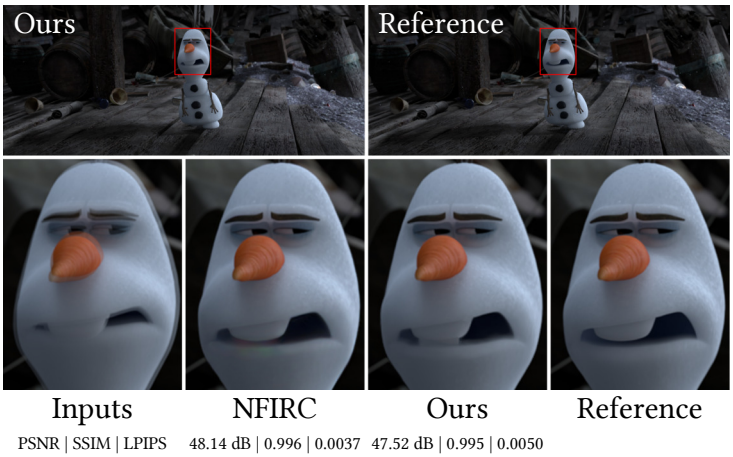

Figure 10: A difficult interpolation example where the mouth opens and closes between the keyframes, thus some of the regions are not present in any of the inputs. Direct prediction attempts to generate some details, but produces slight artifacts. Our kernel-based method inpaints it from neighboring regions. © 2023 Disney

Figure 11: Visual comparison of frames with the lowest PSNR of the sequence, generated by fixed (left) and adaptive (right) interval interpolation, both using the rendering budget of 20%. The plot (middle) shows PSNR of every frame after applying a 3-element minimum filter. © 2023 Disney / Pixar