


# Practical temporal and stereoscopic filtering for real-time ray tracing

Henrik Philippi<sup>1,2</sup>, Jeppe Revall Frisvad<sup>2</sup> , and Henrik Wann Jensen<sup>1</sup>

<sup>1</sup>Luxion, Denmark

<sup>2</sup>Technical University of Denmark

---

## Abstract

We present a practical method for temporal and stereoscopic filtering that generates stereo-consistent rendering. Existing methods for stereoscopic rendering often reuse samples from one eye for the other or do averaging between the two eyes. These approaches fail in the presence of ray tracing effects such as specular reflections and refractions. We derive a new blending strategy that leverages variance to compute per pixel blending weights for both temporal and stereoscopic rendering. In the temporal domain, our method works well in a low noise context and is robust in the presence of inconsistent motion vectors, where existing methods such as temporal anti-aliasing (TAA) and deep learning super sampling (DLSS) produce artifacts. In the stereoscopic domain, our method provides a new way to ensure consistency between the left and right eyes. The stereoscopic version of our method can be used with our new temporal method or with existing methods such as DLSS and TAA. In all combinations, it reduces the error and significantly increases the consistency between the eyes making it practical for real-time settings such as virtual reality (VR).

## CCS Concepts

• **Computing methodologies** → **Rendering**; Ray tracing; Antialiasing; Virtual reality;

---

## 1. Introduction

In interactive applications, we can rarely afford enough samples to avoid aliasing or stochastic sampling noise. Filtering and reusing samples between frames is therefore a common strategy to improve image quality. This can happen over time, usually via a temporal exponential moving average, to exploit coherence between consecutive frames. This filtering is crucial to achieve antialiased images when supersampling is unavailable and to reduce flicker artifacts in real-time applications. When rendering in stereo, we can also exploit the similarities between the two rendered views. Besides further reducing noise and aliasing, filtering between the images in a stereo pair improves depth perception as it restores similarities between the eyes that help the human visual system establish correspondence between the eyes.

In temporal antialiasing (TAA), pixel samples are reprojected to the next frame and checked for validity [AH95], and the result from the previous frame is linearly blended with that of the current frame [YNS\*09]. Using a constant blending parameter ( $\alpha$ ) between zero and one (usually  $\alpha = 0.1$ ), the weighting of samples from older frames decreases exponentially. In a straightforward way, we can adapt this to blending between the two images in a stereo pair. Such stereoscopic blending would apply the same process in both directions and blend with a stereoscopic blending parameter  $\beta = 0.5$  [MKJ20]. However, when these two methods are

combined, a constant  $\beta$  is not optimal. Due to the potentially differing temporal filter kernels between both eyes, an optimal choice of  $\beta$  should consider the magnitude of variance in both frames and adjust the blending parameter accordingly. We present a method for better per-pixel adaptive selection of blending parameters.

In steady state conditions, the optimal blending parameter ( $\alpha$ ) is such that each sample gets equal weight, which is one over the history length in the temporal case [YNS\*09, YLS20]. Thus, even in this limiting case of no interaction, a constant  $\alpha$  is suboptimal. When interactivity is introduced, the user can change the scene at any point in time. Out of lack of a better choice, a constant  $\alpha$  is then often selected in combination with a validation process that discards history if significant change is detected. This leads to resampling errors and temporal lag, so adaptive selection of the blending parameter is important. Instead of switching between two constant values for the blending parameter, we formulate the blending parameter as a trade-off between bias and variance and estimate it using image statistics.

While adaptive schemes for selecting a temporal blending parameter exist, they either do this based on various thresholds [YNS\*09, Sal16] or do not take variance into account [SPD18]. To summarize our contribution, we present:

- a principled equation for computing per-pixel blending parameters given estimates of bias and variance,

- a practical algorithm for estimating variance and bias to compute per-pixel blending parameters, and
- use of our algorithm to filter real-time renderings both in the temporal and stereoscopic domains.

In the temporal domain, our method works well in low noise scenarios. Our stereoscopic blending on the other hand provides an improvement for a wide range of noise levels, including stochastic noise due to Monte Carlo rendering. For a more robust estimation of variance and bias in the temporal domain in high-noise scenarios, we refer to the work of Schied et al. [SPD18].

## 2. Related Work

Image reconstruction from Monte Carlo sampling needs to address issues of noise as well as aliasing. When rendering for virtual reality (VR) headsets, we have to render two images which are mostly identical. Reusing samples between the pair of images can reduce rendering work load with little loss of quality [MNV\*21]. Another option is to blend samples between the two views in order to lower noise levels in the case of stochastic Monte Carlo rendering [MKKJ19]. We show how to choose per-pixel blending parameters that minimize error when blending between the two views. By blending the two images, we achieve lower error and better stereo coherence even under noise. This improves depth perception.

Real-time render engines usually cope with undersampling of stochastic effects such as screen space ambient occlusion or Monte Carlo ray tracing with a combination of denoising and antialiasing. Both typically make use of temporal information by blending between the newly generated and the previous reprojected frame. Even though temporal antialiasing can deal with noisy inputs to some degree, the high variance in Monte Carlo rendering leads to flickering artifacts [YLS20]. Solutions such as spatiotemporal variance-guided filtering (SVGF) [SKW\*17] rely on a spatiotemporal filters to reconstruct global illumination and use conventional TAA techniques for geometric features [VKI\*18]. Even neural network implementations of TAA, such as NVIDIA's deep learning super sampling (DLSS), still suffer from image degradation under noise and are usually paired up with a denoiser to pre-filter very noisy inputs [NVI20]. While neural networks show promise in combining the two tasks of antialiasing and denoising, even kernel predicting networks exhibit biases such as energy loss [TLP\*22]. We introduce a principled way of choosing the temporal blending parameter under theoretical assumptions and show applications to TAA and stereoscopic filtering.

TAA implementations usually blend the sample history with each new frame using a constant blending parameter. For perfect convergence, all samples should be weighted with the same weight so variance reduction is limited with a fixed blending parameter. Some implementations keep track of sample counts  $N$  to allow for faster convergence of low sample count regions by setting  $\alpha = \frac{1}{N}$  [WMB19, KIM\*19]. In regions with high sample counts, low values of  $\alpha$  can compromise image quality due to temporal lag and resampling errors. The parameter  $\alpha$  is thus usually clamped to a minimum value and history is validated using G-Buffer information [NSL\*07] and variance clamping [Sal16]. When history validation fails, previous samples are ignored which is equivalent to

setting  $\alpha = 1$ . Other ways to steer  $\alpha$  include heuristics based on motion speed [YNS\*09], estimation of the temporal gradient in a pixel [SPD18] or, in the absence of motion vectors, by alignment quality [HTD21]. We treat  $\alpha$  as a parameter that should be optimized to balance bias and variance so that error is minimized.

## 3. Theory

An optimal blending between two frames is dependent on local variance and the difference between the underlying means, that is, bias. We model the pixels of each of the two frames as independent random variables  $X, Y$ . These combine linearly with blending parameter  $\alpha$  using

$$\text{lerp}(X, Y, \alpha) = (1 - \alpha)X + \alpha Y = X + \alpha(Y - X).$$

The blended result has less variance but is potentially biased since the two random variables might have different underlying means  $\mathbb{E}[X] \neq \mathbb{E}[Y]$ . The choice of  $\alpha$  is thus inherently a trade-off between variance and bias and depends on whether we want to estimate  $\mathbb{E}[X]$  or  $\mathbb{E}[Y]$ . Since rendering with conventional methods like ray tracing and rasterization rely on point sampling, we can model the rendered radiance values as random variables with additive noise, where the noise represents aliasing and, if applicable, Monte Carlo noise. Variance is then the error that arises from undersampling the image and sources of bias are geometry visible in one image but not in the other, view-dependent shading, or errors introduced from resampling.

Suppose  $X, Y$  are radiance estimates  $X = \mathbb{E}[X] + n_X$  and  $Y = \mathbb{E}[Y] + n_Y$  with  $n_X, n_Y$  being zero-mean noise vectors such that  $\text{Var}(\{X, Y\}) = \sigma_{\{X, Y\}}^2 = \mathbb{E}[n_{\{X, Y\}}^2]$ . We want to estimate  $\mathbb{E}[Y]$  with a blend of the two random variables to obtain  $\text{lerp}(X, Y, \alpha) \approx \mathbb{E}[Y]$  with minimal error. The optimal choice for the blending parameter  $\alpha$  with respect to the expected mean squared error is

$$\arg \min_{\alpha} f(\alpha) = \arg \min_{\alpha} (\mathbb{E}[(\text{lerp}(X, Y, \alpha) - \mathbb{E}[Y])^2]). \quad (1)$$

Substituting  $X$  and  $Y$  with their respective mean and noise, another way to write the objective function is

$$f(\alpha) = \mathbb{E}[(\text{lerp}(n_X, n_Y, \alpha) + (1 - \alpha)(\mathbb{E}[X] - \mathbb{E}[Y]))^2]. \quad (2)$$

Since we assume  $X, Y$  independent (so that  $\mathbb{E}[n_X n_Y] = 0$ ) and recalling that the noise vectors have zero mean (so that  $\mathbb{E}[n_X] = \mathbb{E}[n_Y] = 0$ ), we get

$$f(\alpha) = (1 - \alpha)^2 (\mathbb{E}[X] - \mathbb{E}[Y])^2 + \sigma_X^2 + \alpha^2 (\sigma_X^2 + \sigma_Y^2) - 2\alpha \sigma_X^2. \quad (3)$$

Note that this equation has a term dependent on bias and one dependent on the variance of the underlying distribution. Minimizing the sum of the two terms will balance bias and variance. Since  $\text{Bias}^2 = (\mathbb{E}[X] - \mathbb{E}[Y])^2$  and variances are positive, the function is a parabola with a minimum at

$$\alpha' = 1 - \frac{\sigma_Y^2}{\text{Bias}^2 + \sigma_X^2 + \sigma_Y^2} = 1 - \frac{\sigma_Y^2}{\mathbb{E}[(X - Y)^2]}. \quad (4)$$

This formula for the blending parameter is our key theoretical contribution. In later sections, we will put it to work in different scenarios with different techniques for estimating bias and variance. First, as a sanity check, we make sure that our formula produces the expected blending parameters in special cases.

### 3.1. Temporal Domain

While the validity of this theoretical optimum is straightforward when blending once, temporal filtering applies the blending in sequence with each new frame and therefore additional considerations apply. To illustrate how this  $\alpha'$  relates to commonly used blending strategies, we can look at a special case in which  $\text{Bias}^2 = 0$ . Given a temporally filtered and reprojected estimate  $\tilde{X}_{t-1}$  of frame  $t-1$  and a single sample estimate  $X_t$  of frame  $t$  with no camera movement and no changes in the scene, both variables have the same underlying distribution  $(\mathbb{E}[X], \sigma)$ . Variance of the exponentially filtered variable  $\tilde{X}_{t-1}$  is dependent on its effective sample count  $N_{t-1}$ . The effective sample count of an estimate is the reciprocal of the variance reduction factor  $N_t^{-1}$  such that  $\text{Var}(\tilde{X}_t) = \frac{\text{Var}(X)}{N_t}$ . For a blended random variable, it is

$$N(\text{lerp}(X, Y, \alpha)) = \frac{N(X)N(Y)}{\alpha^2 N(X) + (1-\alpha)^2 N(Y)}. \quad (5)$$

A similar equation was derived by Yang et al. [YNS\*09]. Our version is an extended form to include the number of samples of both the random variables as parameters.

Since we know that it is ideal in this case to give each sample equal weight,  $\alpha'$  should reflect this. By plugging in the quantities for variance and mean into Eq. 4, we get

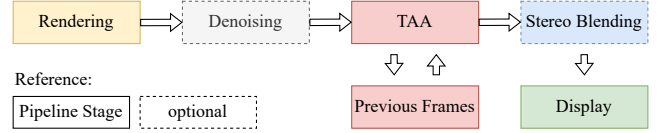
$$\begin{aligned} \alpha' &= 1 - \frac{\text{Var}(X_t)}{(\mathbb{E}[\tilde{X}_{t-1}] - \mathbb{E}[X_t])^2 + \text{Var}(\tilde{X}_{t-1}) + \text{Var}(X_t)} \\ &= 1 - \frac{\sigma^2}{0 + \frac{\sigma^2}{N_{t-1}} + \sigma^2} = \frac{1}{N_{t-1} + 1} = \frac{1}{N_t}, \end{aligned} \quad (6)$$

which is equivalent to progressive rendering with equal weights for each sample and ensures convergence to the true mean.

For the case that  $\text{Bias}^2 \neq 0$ , successive filtering with  $\alpha'$  does not guarantee minimal error since the optimum we find only applies to a single step of filtering. However, we argue that such a global optimum is not practical for two reasons: first, to be useful for real-time rendering, we can only look at a limited number of past frames and have no access to future camera positions and scene changes to predict which samples to keep and which to discard. Secondly, such a global optimum could trade additional temporal lag in one frame for lower variance in another frame leading to inconsistent quality across multiple frames. Furthermore, we can show that, when restricted to only holding a single past frame, no pixel's error can be improved without increasing the error of another, previous pixel's error. This is because in such a setup the error of any pixel can only be reduced by changing a blending parameter of a past frame since the current frame is already at an optimum. However, exchanging this past  $\alpha$  value for a pixel in any of the previous frames moves it away from the optimum and increases the error of that pixel. Therefore, we consider our approach suitable for the temporal domain.

### 3.2. Relationship Between Bias and Gradient

Some heuristics to steer the blending parameter rely on temporal image gradients [SPD18]. The blending parameter is limited in proportion to the relative change  $\frac{\delta}{\mathbb{E}[X]}$  in each frame. Here, we show the relationship between gradient  $\delta$  and Bias in our solution for  $\alpha$ .



**Figure 1:** A typical real-time rendering setup. Temporal antialiasing is applied as a post process to avoid aliasing artifacts and flickering. For stochastic effects such as soft shadows, indirect illumination or specular/glossy reflections, denoising can be employed to spatiotemporally filter out noise. However, some residual noise can still make it into the output image and degrade depth perception and image quality. We add an optional stereo blending stage that reduces error and improves stereo vision in the presence of (residual) noise and aliasing.

Under a constant  $\alpha$ , temporal filtering boils down to a moving exponential average and Bias in a signal of length  $L$  is

$$\begin{aligned} \text{Bias}^2 &= (\mathbb{E}[\tilde{X}_{t-1}] - \mathbb{E}[X_t])^2 \\ &= \left( \mathbb{E}[X_t] - \sum_{i=1}^L (1-\alpha)^{i-1} \alpha \mathbb{E}[X_{t-i}] \right)^2. \end{aligned} \quad (7)$$

If we now assume a constant rate of change for luminance, then  $\mathbb{E}[X_{t-i}] = \mathbb{E}[X_t] - i\delta$ , and we get power series that converge to polynomial functions of orders 0 and  $-1$  for  $L \rightarrow \infty$ :

$$\frac{\alpha}{1-\alpha} \sum_{i=1}^{\infty} (1-\alpha)^i (\mathbb{E}[X_t] - i\delta) = \mathbb{E}[X_t] - \frac{\delta}{\alpha}. \quad (8)$$

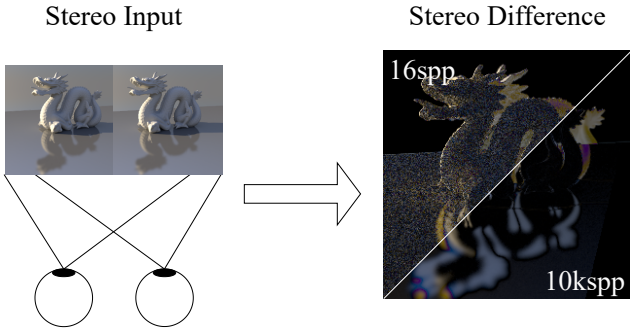
By insertion in Eq. 7, this leads to

$$\text{Bias}^2 = \left( \frac{\delta}{\alpha} \right)^2. \quad (9)$$

Thus, in this case, we have  $\text{Bias}^2$  proportional to  $\delta^2$ . In high sample count scenarios, bias becomes the dominating factor in  $\alpha$  as  $\text{Var}(\tilde{X}_{t-1})$  approaches zero. Enforcing a lower limit based on the image gradient therefore helps prevent bias artifacts like ghosting, temporal lag, and blur but does not necessarily minimize error.

## 4. Application

Because of the general nature of our formula for the blending parameter (Eq. 4), it is potentially useful in many situations where temporal or stereoscopic blending is applied. We focus on real-time rendering and rendering pipelines that follow the structure depicted in Figure 1. Temporal Antialiasing and denoisers integrate into a typical real-time rendering pipeline very similarly in that they take the rendered images and auxiliary data, usually motion vectors, depth and a G-Buffer, and then output a temporally integrated image which is carried over to the next frame. We focus on the post-processing stages of the pipeline and demonstrate the usefulness of our blending parameter for improving stereo vision by combining samples between eyes to better image coherence and reduce residual noise. We also use it to improve validation in a conventional temporal antialiasing context.



**Figure 2:** When presented with a stereo image, depth is perceived by the human visual system by matching the two shown images. We use a reprojected stereo difference as proxy for depth perception. Image regions that differ in the stereo difference are either occluded in one eye or indicate a difference between true and perceived depth such as in the glossy reflection on the floor. At low to moderate sample counts, noise left in the image inhibits depth perception.

#### 4.1. Stereoscopic Filtering

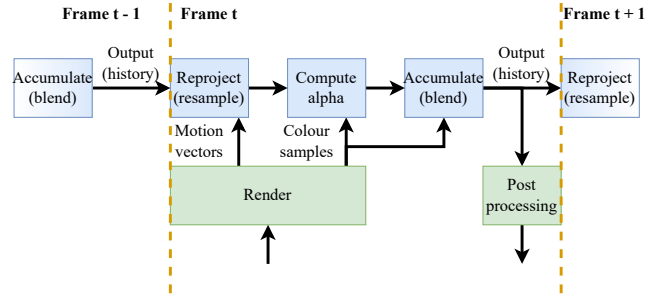
When rendering stochastic effects, residual noise can make it into the final image even when using denoisers. While this residual noise is most of the time low enough not to be perceivable, it can inhibit stereo vision. The correspondences and discrepancies between the images give the human vision system information to perceive depth. Consequently, presenting the eyes with two images with independent noise, regions that should look similar – do not – and regions that should not look similar – do – leading to poor depth perception. As a proxy for the quality of stereo vision, we compute the reprojected squared radiance difference between the two eyes. Figure 2 shows how this is a measure of stereo coherence and therefore depth perception.

An additional benefit from filtering the two images is lowering the rendering work in ray tracing since combining rendering results between eyes has the potential to decrease necessary sample counts and therefore improve performance on top of improving quality. In the stereoscopic context, we call the blending parameter  $\beta$  to distinguish it from the temporal parameter which is applied cumulatively whereas stereo blending is only applied before displaying the final result to the eye and not reused.

Since, we do not have access to reference values of bias and variance, we need to estimate the blending parameter  $\beta$ . We estimate the numerator and the denominator of Eq. 4 separately. The numerator is the variance of the view  $Y$  we are trying to improve, and we estimate it by computing first and second raw moments in a  $3 \times 3$  neighborhood  $\mathcal{N}$  as done in variance clipping [Sal16]:

$$\sigma_Y^2 \approx \frac{1}{|\mathcal{N}|} \left( \sum_{i \in \mathcal{N}} Y_i^2 - \left( \sum_{i \in \mathcal{N}} Y_i \right)^2 \right). \quad (10)$$

For the denominator, we use the same  $3 \times 3$  neighborhood to compute a mean squared difference between the reprojected sample



**Figure 3:** Temporal antialiasing is implemented in three components. Reprojection, Validation and Accumulation. Neural network implementations of TAA such as DLSS and XeSS substitute the validation and accumulation step with a neural network while denoisers are usually put in front of TAA so that they can denoise specular and diffuse components separately. Slight adjustment of a similar figure by Yang et al. [YLS20]

from the other view  $Y$  and each of the samples under the kernel:

$$\mathbb{E}[(X - Y)^2] \approx \frac{1}{|\mathcal{N}|} \sum_{i \in \mathcal{N}} (Y - X_i)^2. \quad (11)$$

This very simple scheme is enough to use our per-pixel adaptive blending parameter for stereo filtering.

#### 4.2. Blending Weights in Temporal Antialiasing

A conventional TAA algorithm has three steps: reprojection, validation and accumulation (see Figure 3). A sample history in the form of a temporally filtered mean  $\bar{X}_{t-1}$  is carried from frame to frame and, in each frame reprojected to the current frame. The reprojected frame is then validated and blended with the new frame  $X_t$  (see also Sec. 3.1). The blending parameter  $\alpha$  is chosen during validation and determines, for each pixel, how much of the sample history we reuse for the current frame. In neighborhood clamping and variance clamping [Sal16], the sample history is either kept and blended with a constant mixing parameter ( $\alpha = c$ , where usually  $c = 0.1$ ) or discarded entirely ( $\alpha = 1$ ).

This either-or approach in which history is either kept or entirely discarded easily leads to image artifacts such as ghosting or flickering and can therefore benefit from the more continuous approach offered by our Eq. 4. Since the  $\alpha$  obtained with our method naturally captures disocclusions, resampling blur and temporal lag at the same time, we can apply it without any heuristics based on G-buffer data or based on velocity. To compute  $\alpha$ , we use the same estimates as in stereoscopic filtering but replace the reprojected other eye's view with the temporal accumulation image which is reprojected into each new frame.

#### 5. Results

We implemented our stereoscopic and temporal filtering into a real-time ray tracing framework based on the GPU ray tracing extensions in Vulkan [KHBW20]. Vulkan also enables combination





**Figure 4:** The three test scenes used for testing TAA performance. First scene is a static view of a fence with its moving shadow. Second scene is a sequence of VR poses in a classroom scene. Third scene is a smooth camera movement through a Minecraft map. The closeups show quality comparisons of our adjusted temporal antialiasing to conventional TAA with neighborhood clamping as well as DLSS. Our method specifically improves moving shadows and fine geometry that is far away.

of our implementation with DLSS (<https://developer.nvidia.com/-rtx/dlss>). Performance is measured in headless so without presenting an image to a window or head-mounted display. Image quality measurements are done in linear RGB space. All performance results were measured on an RTX 3090 GPU.

### 5.1. Validation in Temporal Antialiasing

We compare our method (ours) to a standard neighborhood clamping TAA (taa) as well as DLSS 3.1 at 100% render resolution with frame generation turned off (dlss). We leave out any history rectification in taa and ours since it worsens objective error measurements and improving rectification is orthogonal to this paper. We measure frame times and MSE for the three scenes shown in Figure 4 (top). The first scene fence has a static camera with a fence casting a shadow from a moving sun. The scene highlights a problematic situation for standard TAA and DLSS as the moving shadow is challenging to antialias because of false motion vectors and many jagged edges. The second scene classroom is a sequence of prerecorded VR poses, that is, positions and orientations of a VR

headset in a classroom scene. We use real tracking data of a VR headset because the noisy nature of the resulting camera views can be challenging for temporal integration algorithms. Thirdly, we use a smooth camera path through a Minecraft map rungholt which contains some distant details as well as textures that require good antialiasing. The render resolutions are  $1024 \times 1024$  for fence,  $1440 \times 1600 \times 2$  for classroom, which is a standard VR headset resolution, and  $1920 \times 1080$  for rungholt.

**Image quality.** The following table shows resulting RMSE and SSIM scores for the complete image sequences. The column called 1spp is a one sample per pixel baseline. Arrows indicate whether a better value is lower ( $\downarrow$ ) or higher ( $\uparrow$ ).

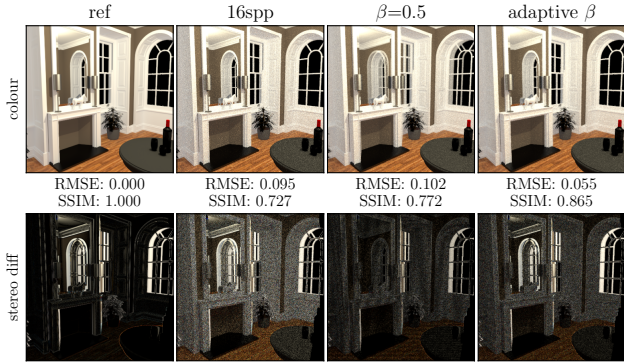
scene		1spp	dlss	taa	ours
fence	RMSE	$\downarrow$ 0.015	0.009	0.009	<b>0.006</b>
	SSIM	$\uparrow$ 0.990	0.991	0.988	<b>0.997</b>
classroom	RMSE	$\downarrow$ 0.074	0.060	0.048	<b>0.037</b>
	SSIM	$\uparrow$ 0.960	0.983	0.985	<b>0.987</b>
rungholt	RMSE	$\downarrow$ 0.050	0.033	0.032	<b>0.029</b>
	SSIM	$\uparrow$ 0.921	0.964	0.967	<b>0.970</b>

Our method performs well across the board. Figure 4 highlights some situations in which our method performs well. The difference is most visible in the fence scene where the other methods exhibit ghosting and/or aliasing. Our method improves on this and keeps historical samples partially but attenuates their contribution which leads to some antialiasing and error reduction even in changing regions unlike the result of a complete acceptance/rejection of the history. The moving shadows in the classroom scene also benefit slightly from our method although DLSS leads to a visually more pleasing albeit less correct result. The rungholt scene shows a second situation in which other methods fail. The far away ship contains fine geometry that leads to samples being rejected inconsistently by DLSS and neighborhood clamping. This leads to jittering motion (see supplemental video) and excessive blur. Our method fixes these artifacts mostly and exhibits no jittering motion.

**Performance.** All three methods are light-weight post-processing passes that need less than 1 ms of execution time at typical rendering resolutions. In the following, we present overhead in milliseconds as well as 95% confidence intervals ( $\pm$ ) for the three methods. The 720p, 1080p and 4k resolutions are common render resolutions for traditional displays and CV1( $1080 \times 1200 \times 2$ ), Rift S( $1280 \times 1440 \times 2$ ) and Quest 1( $1440 \times 1600 \times 2$ ) are resolutions of common head mounted displays:

Resolution	dlss	taa	ours
720p	0.26( $\pm 0.00$ )	0.12( $\pm 0.04$ )	0.10( $\pm 0.00$ )
1080p	0.41( $\pm 0.05$ )	0.25( $\pm 0.06$ )	0.22( $\pm 0.00$ )
4k	1.22( $\pm 0.16$ )	0.96( $\pm 0.16$ )	0.90( $\pm 0.10$ )
CV1	0.64( $\pm 0.10$ )	0.30( $\pm 0.06$ )	0.28( $\pm 0.04$ )
Rift S	0.77( $\pm 0.11$ )	0.43( $\pm 0.08$ )	0.44( $\pm 0.12$ )
Quest 1	0.85( $\pm 0.09$ )	0.54( $\pm 0.12$ )	0.49( $\pm 0.06$ )

DLSS has the highest overhead although this is without any up-scaling, which would improve overall performance significantly but also hurt image quality. Since ours requires no additional data



**Figure 5:** Quality comparison between a reference (ref), a 16spp rendered image and two strategies for stereo blending the 16spp image. The bottom row shows the reprojected stereo difference explained in Figure 2. A fixed  $\beta$  parameter leads to double vision and false sense of depth in specular materials as well as misplaced highlights on glossy materials. The introduced bias also worsens error. These situations would have to be fixed by bespoke heuristics based on auxiliary data. Our adaptive  $\beta$  parameter deals with all these situations in a single equation that leads to lower error while leaving depth perception intact.

compared to taa, performance is equivalent with any measured differences coming down to minor implementation differences.

## 5.2. Stereoscopic Filtering

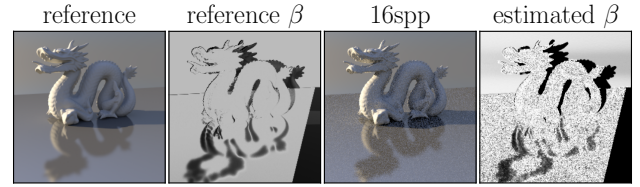
Our stereoscopic filtering pass is an additional post-processing after temporal filtering. The implementation of this is almost identical to the temporal antialiasing filter. Just like the temporal version, inputs are depth and colour buffers and the reprojection and filtering is done at once. The difference is that the input images are the left and right eye or vice versa instead of the new colour image and the accumulation image. We add an additional depth check to reject reprojected samples from disoccluded regions.

**Image quality.** The improvement in image quality can be seen in Figure 5. Stereoscopic filtering helps lower overall error without introducing significant bias. In terms of stereo difference, the improvement is significant

**Performance.** The overhead of stereo blending is dependent on the rendering resolution and listed in the following table. We measure overhead in milliseconds for some common VR headset resolutions. Just like TAA, the overhead is small compared to other render tasks in a render pipeline.

Resolution	Stereo Blending
1080 × 1200 × 2 (CV1)	0.31(±0.12)
1280 × 1440 × 2 (Rift S)	0.38(±0.09)
1440 × 1600 × 2 (Quest 1)	0.48(±0.11)

**Comparison to reference blending.** By computing ground truth values for bias and variance, we can obtain per-pixel reference values for  $\beta$ . For a selected number of samples per pixel, we pre-



**Figure 6:** Comparison of our estimate of  $\beta$  versus reference  $\beta$  for a 16spp image. The reference  $\beta$  was computed using reference values for reprojection bias and variances of the two stereo views.

compute reference values for the first and second raw moments ( $\mathbb{E}(\{X, Y\})$  and  $\mathbb{E}(\{X^2, Y^2\})$ ) of the radiance as well as stereo disparity vectors for reprojection. With these we can find the reference bias and variances needed for Eq. 4. We compare the reference values to our estimated  $\beta$  value in Figure 6. We observe that on average  $\beta$  is higher in our estimate which is likely because the resampling step blurs reprojected samples and therefore lowers its variance.

## 5.3. Combined Results

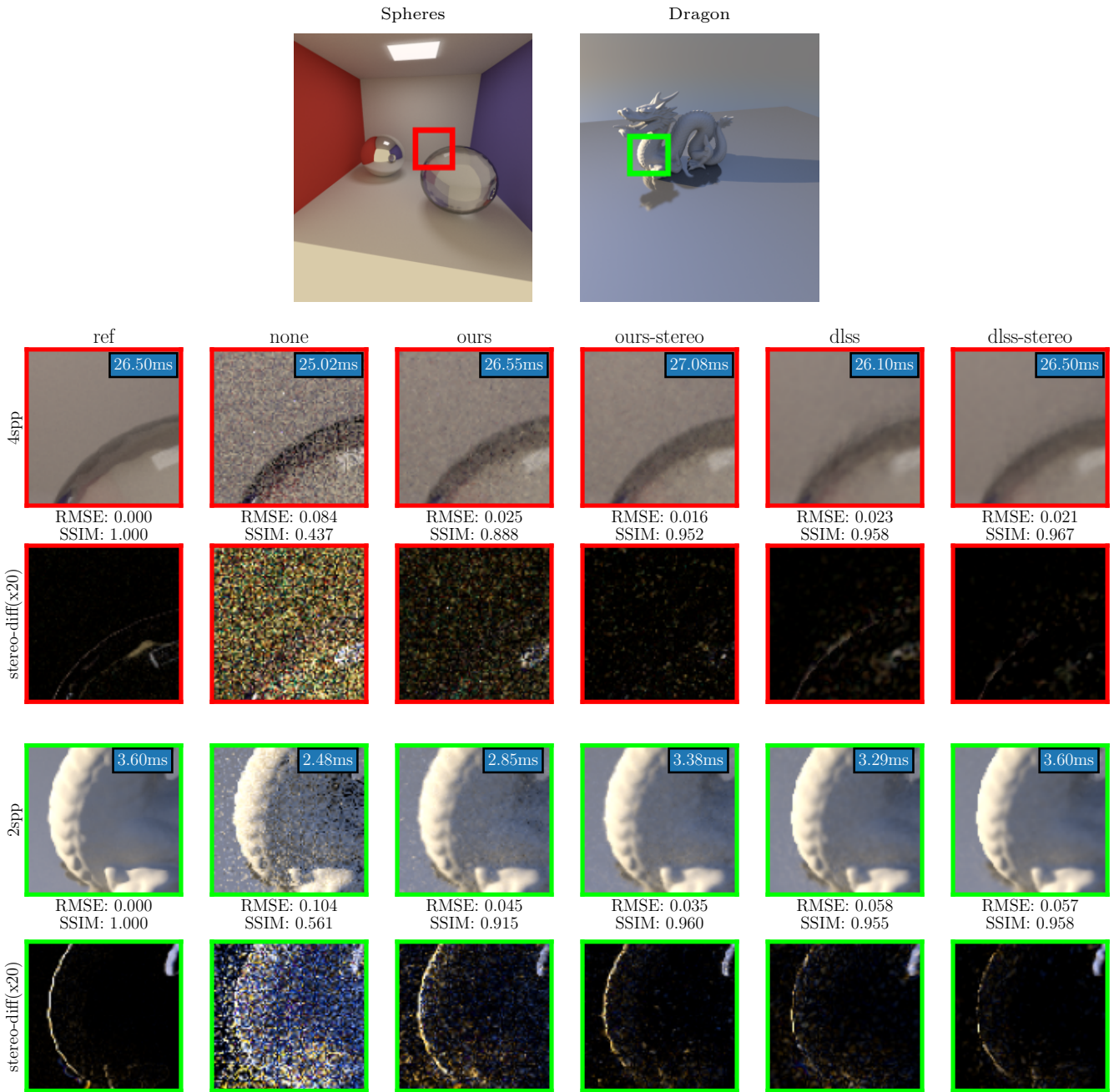
In practice, temporal filtering and stereoscopic would and should be combined to get the best results for real-time rendering (see Figure 1). While our temporal filter is superior in non-noisy scenes, correct estimation of  $\alpha$  is challenging in scenes with noisy effects from path tracing, for example. We therefore show combined results for video sequences rendered with our stereoscopic filtering on top of our temporal filtering as well as in combination with DLSS, where DLSS takes the place of a TAA implementation. We apply no denoising and let DLSS or our temporal filter handle the noise. We use two scenes with stochastic effects as test. One is a CornellBox with specular spheres and the other is the Stanford Dragon on top of a glossy plane. The error measurements are summarized in the following table.

scene		none	ours	ours stereo	dlss	dlss stereo
Spheres	RMSE ↓	0.071	0.023	<b>0.016</b>	0.017	<b>0.016</b>
	SSIM ↑	0.591	0.910	0.958	0.978	<b>0.982</b>
Dragon	RMSE ↓	0.046	0.018	<b>0.017</b>	0.020	0.021
	SSIM ↑	0.749	0.960	0.980	0.985	<b>0.988</b>

Figure 7 shows some closeups including average frame times for each sequence. Stereo blending does best in freshly disoccluded regions that were already visible earlier in the other eye. In addition to better error when combined with our temporal filter, we highlight the fact that stereo blending improves depth perception in all scenarios. The presented stereo differences show how stereo vision is improved and closer to the reference across the board. We also attach a video that demonstrates the effect on stereo vision using the dragon scene.

## 6. Discussion and Conclusion

We proposed a principled way to determine temporal and stereo blending parameters and demonstrated its effectiveness in real-time



**Figure 7:** Results for our combined stereoscopic and temporal blending. Stereoscopic blending improves error when used in combination with our temporal filter. In combination with DLSS, error reduction is low, but stereo coherence is greatly improved. The full sequence is attached as a video in the additional materials.

rendering. The temporal parameter is easily integrated in currently available TAA solutions while stereo blending only requires an additional lightweight post-processing step. The key equation (Eq. 4) is applicable to a wide range of scenarios where two estimates are combined to retrieve a better estimate and we present two practical

examples of how to integrate it into an existing real-time rendering pipeline with little overhead.

**Limitations and future work.** The characteristics of human depth perception when presented with noisy, Monte Carlo rendered images seem mostly unknown and strategies for improving it would



benefit from knowledge about its underlying mechanisms. We presented a proxy for stereo vision in the form of reprojected stereo differences, this approach is however limited when specular reflections are involved. Similarly, our method is unable to improve stereo vision when no stereo discrepancy vectors are available, but recent work has made some advancements in these situations [ZLY\*21, HTD21]. In theory, our temporal filtering method could be used for denoising of highly noisy signals such as those obtained with path tracing. In practice, however, our estimates of bias and variance were too noisy. We leave the research to find better estimates for future work. Since all denoisers have to do the same compromise between bias and variance that we describe here, future research should benefit from the ability of our method to provide reference values for the temporal blending parameters.

**Acknowledgements** This project has received funding from the European Union's Horizon 2020 research and innovation programme under the Marie Skłodowska-Curie grant agreement No 956585 (PRIME). Credit for the used scenes go to their respective authors. Dragon, Rungholt, Cornellbox and Fireplace are from McGuire Computer Graphics Archive [McG17]. The classroom scene is from the Blender Foundation's website.

## References

- [AH95] ADELSON S. J., HODGES L. F.: Generating exact ray-traced animation frames by reprojection. *IEEE Computer Graphics and Applications* 15, 3 (1995), 43–52. doi:10.1109/38.376612. 1
- [HTD21] HANIKA J., TESSARI L., DACHSBACHER C.: Fast temporal reprojection without motion vectors. *Journal of Computer Graphics Techniques* 10, 3 (September 2021), 19–45. URL: <http://jcgtag.org/published/0010/03/02/>. 2, 8
- [KHBW20] KOCH D., HECTOR T., BARCZAK J., WERNESSE E.: Ray tracing in Vulkan. Khronos Blog, March 2020. URL: <https://www.khronos.org/blog/ray-tracing-in-vulkan>. 4
- [KIM\*19] KOSKELA M., IMMONEN K., MÄKITALO M., FOI A., VIITANEN T., JÄÄSKELÄINEN P., KULTALA H., TAKALA J.: Block-wise multi-order feature regression for real-time path-tracing reconstruction. *ACM Transactions on Graphics* 38, 5 (June 2019). doi:10.1145/3269978. 2
- [McG17] MCGUIRE M.: Computer graphics archive, July 2017. <https://casual-effects.com/data>. URL: <https://casual-effects.com/data>. 8
- [MKJ20] MÄKITALO M. J., KIVI P. E. J., JÄÄSKELÄINEN P. O.: Systematic evaluation of the quality benefits of spatiotemporal sample reprojection in real-time stereoscopic path tracing. *IEEE Access* 8 (2020), 133514–133526. doi:10.1109/ACCESS.2020.3010452. 1
- [MKKJ19] MÄKITALO M., KIVI P., KOSKELA M., JÄÄSKELÄINEN P.: Reducing computational complexity of real-time stereoscopic ray tracing with spatiotemporal sample reprojection. In *Proceedings of the 14th International Joint Conference on Computer Vision, Imaging and Computer Graphics Theory and Applications - GRAPP*, (2019), INSTICC, SciTePress, pp. 367–374. doi:10.5220/0007692103670374. 2
- [MNV\*21] MUELLER J. H., NEFF T., VOGLREITER P., STEINBERGER M., SCHMALSTIEG D.: Temporally adaptive shading reuse for real-time rendering and virtual reality. *ACM Transactions on Graphics* 40, 2 (April 2021), 11:1–11:14. doi:10.1145/3446790. 2
- [NSL\*07] NEHAB D., SANDER P. V., LAWRENCE J., TATARCHUK N., ISIDORO J. R.: Accelerating real-time shading with reverse reprojection caching. In *SIGGRAPH/Eurographics Workshop on Graphics Hardware* (2007), Eurographics Association, pp. 25–35. doi:10.2312/EGGH/EGGH07/025-036. 2
- [NVI20] NVIDIA CORPORATION: NRD Sample: NVIDIA real-time denoisers sample source code, 2020. URL: <https://github.com/NVIDIAGameWorks/NRDSample>. 2
- [Sal16] SALVI M.: An excursion in temporal supersampling. GDC16, 2016. URL: [https://developer.download.nvidia.com/gameworks/events/GDC2016/msalvi\\_temporal\\_supersampling.pdf](https://developer.download.nvidia.com/gameworks/events/GDC2016/msalvi_temporal_supersampling.pdf). 1, 2, 4
- [SKW\*17] SCHIED C., KAPLANYAN A., WYMAN C., PATNEY A., CHAITANYA C. R. A., BURGESS J., LIU S., DACHSBACHER C., LEFOHN A., SALVI M.: Spatiotemporal variance-guided filtering: Real-time reconstruction for path-traced global illumination. In *High Performance Graphics (HPG)* (2017), pp. 2:1–2:12. doi:10.1145/3105762.3105770. 2
- [SPD18] SCHIED C., PETERS C., DACHSBACHER C.: Gradient estimation for real-time adaptive temporal filtering. *Proceedings of the ACM on Computer Graphics and Interactive Techniques* 1, 2 (August 2018), 24:1–24:16. doi:10.1145/3233301. 1, 2, 3
- [TLP\*22] THOMAS M. M., LIKTOR G., PETERS C., KIM S., VAIDYANATHAN K., FORBES A. G.: Temporally stable real-time joint neural denoising and supersampling. *Proceedings of the ACM on Computer Graphics and Interactive Techniques* 5, 3 (July 2022), 21:1–21:22. doi:10.1145/3543870. 2
- [VKI\*18] VIITANEN T., KOSKELA M., IMMONEN K., MÄKITALO M., JÄÄSKELÄINEN P., TAKALA J.: Sparse sampling for real-time ray tracing. In *Proceedings of the 13th International Joint Conference on Computer Vision, Imaging and Computer Graphics Theory and Applications - GRAPP*, (2018), INSTICC, SciTePress, pp. 295–302. doi:10.5220/0006655802950302. 2
- [WMB19] WILLBERGER T., MUSTERLE C., BERGMANN S.: Deferred hybrid path tracing. In *Ray Tracing Gems: High-Quality and Real-Time Rendering with DXR and Other APIs*, Haines E., Akenine-Möller T., (Eds.). Apress, Berkeley, CA, 2019, pp. 475–492. doi:10.1007/978-1-4842-4427-2\_26. 2
- [YLS20] YANG L., LIU S., SALVI M.: A survey of temporal antialiasing techniques. *Computer Graphics Forum* 39, 2 (2020), 607–621. doi:10.1111/cgf.14018. 1, 2, 4
- [YNS\*09] YANG L., NEHAB D., SANDER P. V., SITTHI-AMORN P., LAWRENCE J., HOPPE H.: Amortized supersampling. *ACM Transactions on Graphics* 28, 5 (December 2009), 1–12. doi:10.1145/1618452.1618481. 1, 2, 3
- [ZLY\*21] ZENG Z., LIU S., YANG J., WANG L., YAN L.-Q.: Temporally reliable motion vectors for real-time ray tracing. *Computer Graphics Forum* 40, 2 (2021), 79–90. doi:10.1111/cgf.142616. 8