Stereo-consistent screen-space ambient occlusion

PEITENG SHI, Delft University of Technology, the Netherlands and National University of Defense Technology, China
MARKUS BILLETER, University of Leeds, United Kingdom
ELMAR EISEMANN, Delft University of Technology, the Netherlands

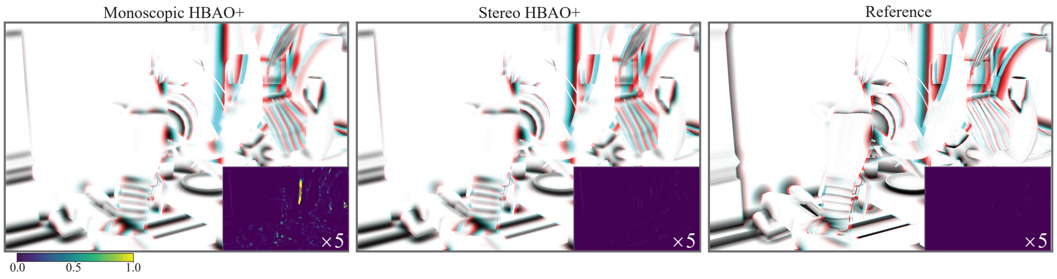

Fig. 1. Monoscopic HBAO+ generates inconsistent stereo images (left). Our stereo HBAO+ (middle) reduces the inconsistency errors close to that of a geometry-based pre-computed reference solution (right). Please view the images zoomed-in with anaglyph glasses. Inconsistency errors (5 × magnification), as defined in our evaluation, are shown as insets in the bottom right. The model is from the Sea House scene by Alena Shek. The model is licensed under CC BY 4.0 .

Screen-space ambient occlusion (SSAO) shows high efficiency and is widely used in real-time 3D applications. However, using SSAO algorithms in stereo rendering can lead to inconsistencies due to the differences in the screen-space information captured by the left and right eye. This will affect the perception of the scene and may be a source of viewer discomfort. In this paper, we show that the raw obscurance estimation part and subsequent filtering are both sources of inconsistencies. We developed a screen-space method involving both views in conjunction, leading to a stereo-aware raw obscurance estimation method and a stereo-aware bilateral filter. The results show that our method reduces stereo inconsistencies to a level comparable to geometry-based AO solutions, while maintaining the performance benefits of a screen-space approach.

CCS Concepts: • **Computing methodologies** → **Rendering**; **Rasterization**.

Additional Key Words and Phrases: screen-space ambient occlusion, stereo consistency, VR

ACM Reference Format:
Peiteng Shi, Markus Billeter, and Elmar Eisemann. 2022. Stereo-consistent screen-space ambient occlusion. *Proc. ACM Comput. Graph. Interact. Tech.* 5, 1, Article 2 (May 2022), 12 pages. https://doi.org/10.1145/3522614

Authors' addresses: Peiteng Shi, p.shi@tudelft.nl, Delft University of Technology, Van Mourik Broekmanweg 6, Delft, the Netherlands, 2628 XE and National University of Defense Technology, Deya Road, Delft, China, 410073; Markus Billeter, M.Billeter@leeds.ac.uk, University of Leeds, Woodhouse Lane, Leeds, United Kingdom, LS2 9JT; Elmar Eisemann, p.shi@tudelft.nl, Delft University of Technology, Van Mourik Broekmanweg 6, Delft, the Netherlands, 2628 XE.

2577-6193/2022/5-ART2
https://doi.org/10.1145/3522614

1 INTRODUCTION

With the increasing presence of Virtual Reality (VR) headsets in the market, stereo content is becoming popular and receives much attention. Efficient graphics algorithms to synthesize high-quality imagery play a key role in providing an immersive experience. Ambient occlusion (AO) is one of them, which is an illumination effect that can be used to render high-quality images. However, computing AO using geometry is very expensive. Hence, much research investigates real-time AO in screen space.

While geometry-based AO is view-independent, fast screen-space methods are not. The reason is that the captured scene geometry in screen space might vary between different camera views. In consequence, screen-space AO (SSAO) methods can introduce an inconsistency between stereo image pairs, leading to a potential source of discomfort, which can also interfere with depth perception [Akenine-Möller et al. 2018; Bukenberger et al. 2018; Gong et al. 2018; Kooi and Toet 2004]. Extending SSAO to stereo is an important goal to ensure an immersive experience.

In this paper, we reveal two major sources for inconsistencies of conventional SSAO algorithms, which relate to the raw obscurance estimation and the bilateral filter pass. Our main contribution is to make both of these steps stereo-aware. We achieve this goal by allowing a given pixel to access screen-space geometry in both views simultaneously. Figure 1 presents one example showing that our method reduces the inconsistency errors of the monoscopic SSAO to a level that approaches the quality of geometry-based AO.

2 RELATED RESEARCH

Ambient occlusion encodes how ambient illumination arrives at a point that is locally blocked by nearby objects [Zhukov et al. 1998]. AO can hereby greatly enhance the realism of a scene [Mattausch et al. 2010]. As indicated, there are two major choices for the computation space: geometry-based and screen space-based methods [McGuire et al. 2011]. In this paper, we will mainly focus on real-time rendering, the interested reader is referred to an overview of additional AO research [Akenine-Möller et al. 2018].

In order to achieve real-time performance in geometry-based AO computation, a spatial hashing data structure [Gautron 2020] can speed up the computation of ray-traced AO. The concept of occlusion volumes [McGuire 2010] is similar to shadow volumes [Crow 1977] but occluders influence their surrounding, reaching the quality similar to ray-traced AO. The method still suffers from over-draw, since the occlusion volume of each polygon has to be rasterized. Computing AO based on a scene's source geometry ensures that the AO results are independent of camera views, thus consistent in stereo images. However, the performance of the previous AO algorithms highly depends on the scene's complexity (e.g., the number of triangles). Further, the performance of geometry-based AO algorithms is usually much lower than screen-space alternatives [Vermeer et al. 2021].

Screen-space AO uses the depth map (and sometimes also the normal map) as an approximation of the scene geometry. AO computations then only depend on the resolution of the rendered images instead of the geometric scene complexity [Mittring 2007; Shanmugam and Arikan 2007]. The original SSAO approach [Mittring 2007] generates 3D random samples in a hemisphere to evaluate the obscurance for a certain point in an image. VAO [Loos and Sloan 2010] replaces the 3D samples of the original SSAO into line samples to reduce the required number of random samples. Bavoil [Bavoil et al. 2008] presented Horizon-Based AO algorithm (HBAO), which estimates the maximum horizon angle exploring several directions. McGuire [McGuire et al. 2011] proposed Alchemy Ambient Obscurance (Alchemy AO), which takes the horizon angle and distance of the samples from the shaded point into account. HBAO+ [NVIDIA 2016] generates 2D random samples

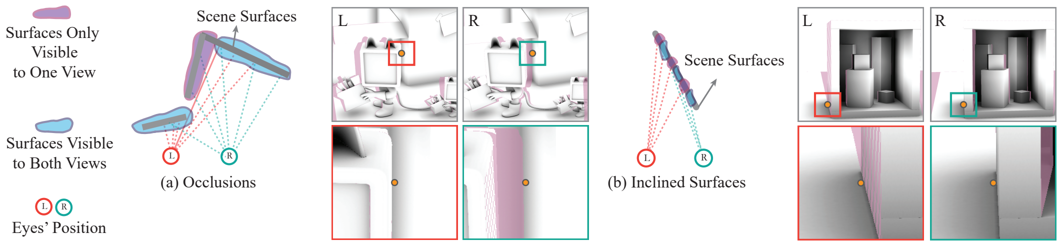

Fig. 2. Screen-space differences between the left eye and the right eye. Purple pixels are only visible to one eye. (a): Some surfaces are only visible for one eye due to occlusions of other surfaces. (b): One camera may only see limited parts of a surface when the surface is inclined towards an eye. The yellow point represents a shaded point that is visible in both views. The surrounding screen-space scene geometry of the shaded point is different (highlighted region in (a) and (b)). The Robot cat scene is modified from the Junk Shop scene (the electronic version contains hyperlinks to the models). This Blender demo scene is made by Alex Treviño and licensed under CC BY .

like HBAO but uses the same raw obscurance estimator as Alchemy AO. The implementation details of HBAO+ are also summarized by Vermeer et al. [Vermeer et al. 2021]. Jorge et al. [Jiménez et al. 2016] extend HBAO by incorporating a radiometrically-correct formulation of the ambient occlusion equation.

Although screen-space AO is very efficient, it suffers from problems due to hidden geometry, as only the visible scene surfaces can be considered. To capture some of the otherwise hidden geometry, a two-layer G-Buffer [Mara et al. 2016] or several virtual cameras [Vardis et al. 2013] can be used. The latter computes AO as a weighted average between the main camera view and additional virtual views. Stochastic-Depth AO [Vermeer et al. 2021] records multiple scene layers randomly per pixel, thus, fragments behind the nearest visible surfaces are made available via a multi-sampled depth texture.

As shown by Figure 2, due to occlusions or inclined surfaces, inconsistent AO estimates might arise when evaluating the image of each eye separately. In our paper, we aim at reducing the inconsistency caused by employing screen-space ambient occlusion solutions for stereoscopic views. We base our stereo-aware method on HBAO+, but other SSAO algorithms could be adapted similarly. Hereby, we follow a recent trend towards stereo-consistent algorithms, previously employed to stylization [Northam et al. 2012], style transfer [Gong et al. 2018], and line drawings [Bukenberger et al. 2018; He et al. 2019].

3 STEREO-CONSISTENT SSAO

SSAO methods typically consist of two screen-space passes. The first pass computes a noisy AO estimate and the second filters the results to eliminate the noise. Our method follows this general pattern and proposes to make both passes stereo-aware, as to avoid stereo inconsistencies. In our solution, we evaluate the first pass only for one eye and then project the results to the other eye. This requires us to fill holes after the projection, but it generally results in an improved performance. Our proposed method guarantees consistent results.

3.1 Stereo-aware obscurance estimation

The first pass estimates AO from screen-space information. The estimate must be consistent with both views. We guarantee this by ensuring that the same estimate is produced regardless of the view a certain pixel originates from. Conceptually, when estimating obscurance $O(\mathbf{p})$ for a pixel $\mathbf{p}$

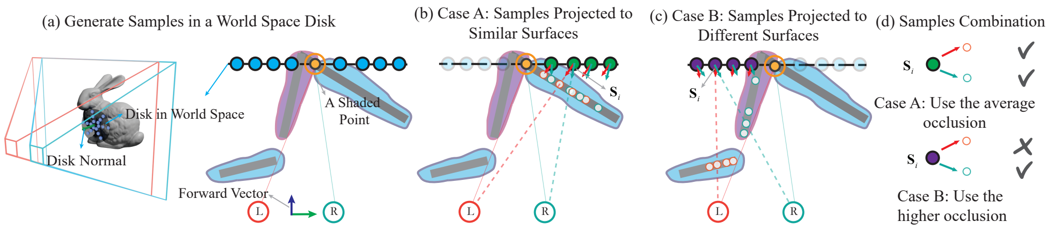
(a) Generate Samples in a World Space Disk
Disk in World Space
Disk Normal
Forward Vector
A Shaded Point
L
R
(b) Case A: Samples Projected to Similar Surfaces
S_i
(c) Case B: Samples Projected to Different Surfaces
(d) Samples Combination
Case A: Use the average occlusion
Case B: Use the higher occlusion

Fig. 3. Stereo obscurance estimation. (a): Samples are generated in a world space disk. Scene surfaces are classified into two categories as shown in Figure 2. (b): Samples that are projected to surfaces visible to both eyes are actually sampling similar surfaces. (c): Samples that are projected to surfaces only visible to one view are sampling two distinct surfaces. (d): We evaluate obscurance for each sample in both views. We use the average obscurance of the two samples in Case A, but choose the higher obscurance value from the two samples for Case B. The Stanford Bunny model made by Stanford University Computer Graphics Laboratory (Stanford Scan license) is from the Computer Graphics Archive [McGuire 2017].

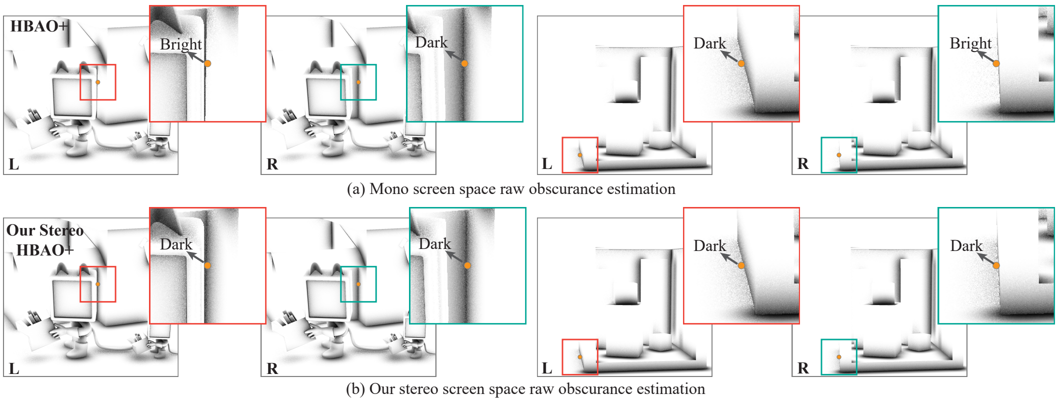
HBAO+
Bright
Dark
Dark
Bright
L
R
L
R
(a) Mono screen space raw obscurance estimation
Our Stereo HBAO+
Dark
Dark
Dark
Dark
L
R
L
R
(b) Our stereo screen space raw obscurance estimation

Fig. 4. Obscurance estimation results comparisons. The yellow dot shows the same position in the left and right views. (a): Obscurance estimation with mono HBAO+. In the left example, the pixels around the highlighted point in the left view are bright as the occluders are not visible in the left view. In the right example, the pixels around the highlighted point are bright in the right view since a very limited area of the wall is visible (also see Figure 2). (b): Our stereo HBAO+ obscurance estimation. The inconsistent estimations in (a) are consistent thanks to taking both views into account during obscurance estimation.

in either view, we find its corresponding position in world space, **P** and express the estimate as a function of this world space position.

To estimate AO, we select a number of samples and then accumulate the obscurance contribution of each sample. Following HBAO/HBAO+ methods[Bavoil et al. 2008; NVIDIA 2016], our samples $\{\mathbf{S}_i\}$ are generated on a disk. In our method, the disk is centered at **P** and aligned with the shared forward vector of the two views (see Figure 3(a-b)). Sample generation schemes of other screen-space AO methods could similarly be adapted.

We evaluate the contribution of a sample $\mathbf{S}_i$ by projecting it into each view, where we compute the impact using the estimator defined in the HBAO+ method. We end up with two contributions for each sample, and must subsequently combine these. Here, we account for two cases. In the first case, where the sample $\mathbf{S}_i$ falls on a location that is visible in both views, we average the two estimates (Case A in Figure 3(b)). Even if it is true that two pixels in both views are unlikely to relate to the exact same world position, they will query similar neighborhoods in world space. In the second case, we pick the maximum obscurance, which accounts for occluders visible in only one of the views (Case B in Figure 3(c)). Figure 3(d) summarizes our samples combination method.

To check whether a sample is visible in both views and to benefit from computational coherence, we perform this test based on a pair of binary textures; $\mathcal{L}^\ell$ and $\mathcal{L}^r$ (for the left and right view, respectively). Each $\mathcal{L}$ encodes whether the point corresponding to a pixel is visible in the respective other view, which is determined by reverse reprojection (RP) [Bavoil and Andersson 2012; Mattausch et al. 2010; Nehab et al. 2007]. A pixel $\mathbf{p}$ is projected to the other view, and the nearest matching pixel $\mathbf{p}^*$ is found in screen space. $\mathbf{p}^*$ is then projected back to the original view. If its location is sufficiently close (1 pixel width), we consider the point visible in both views and mark $\mathbf{p}$ as white. We refer to this process as a *back-check*. In practice, these binary textures are derived directly from the two depth buffers and it is not required to redraw the entire scene. The resulting binary maps are not stored in separate textures, instead, we can encode them in the sign bit of the 32-bit float depth buffers.

As demonstrated in Figure 4, the monocular HBAO+ will generate inconsistent raw obscurance estimates due to differences in the two views (Figure 2). Our stereo-aware method avoids this by taking the screen-space information from both views into account.

Although a binary decision is made per sample, the obscurance estimation used during shading is an integration over many samples. For this reason, the algorithm shows, similar to standard SSAO solutions, a strong temporal stability (as is illustrated in our supplementary video).

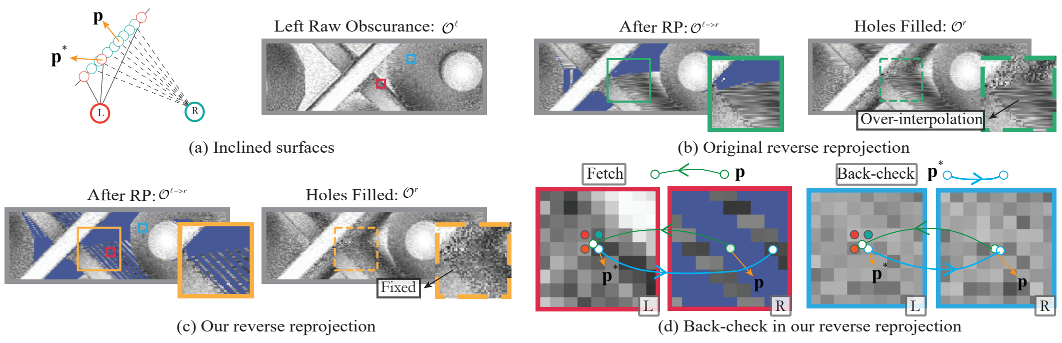
p
p*
L
R
Left Raw Obscurance: $\mathcal{O}^\ell$
After RP: $\mathcal{O}^{\ell\to r}$
Holes Filled: $\mathcal{O}^r$
Over-interpolation
(a) Inclined surfaces
(b) Original reverse reprojection
After RP: $\mathcal{O}^{\ell\to r}$
Holes Filled: $\mathcal{O}^r$
Fixed
Fetch
Back-check
(c) Our reverse reprojection
(d) Back-check in our reverse reprojection

Fig. 5. Reverse reprojection (RP) with back-check. (a): When the surface is unevenly inclined to both cameras, one camera may only see very limited parts of the surfaces. (b): Simply performing the original RP will result in over-interpolation. (c): Our RP with a back-check fixes the over-interpolation. (d): The back-check projects a pixel to the other view and finds the closest pixel there. The pixel is then projected back. If this point is too far from the original pixel, it is rejected as invalid (left pair). If the point is close, the obscurance value is reprojected (right pair). The model is from the Bistro scene [McGuire 2017], which is made by Amazon Lumberyard and licensed under CC BY 4.0 .

3.2 Reprojecting obscurance

Our computations are more expensive than a monocular evaluation, as both views are concerned. Yet, they are also symmetric in both views, which allows us to only compute one view and reproject the result to the whole stereo pair. Naive reprojection [Bavoil and Andersson 2012; Mattausch et al. 2010; Nehab et al. 2007] results in over-interpolation, i.e., where a single pixel's AO estimate is replicated across multiple pixels in the other view, yielding low-frequency artifacts (Figure 5), which cannot be removed by filtering (applied in the next step). We solve this problem by performing

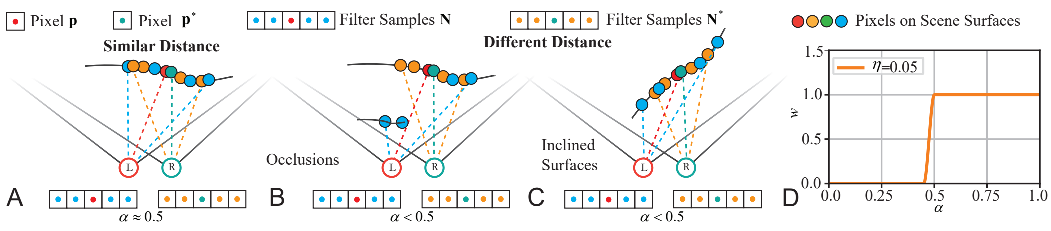

Fig. 6. A: The left filter samples (N) and the right filter samples (N*) of pixel **p** have similar distance to **P**. B-C: Due to occlusions or inclined surfaces, (N) and (N*) have different distance to **P**. D: Plots of the weight w used when combining filtered values from each view.

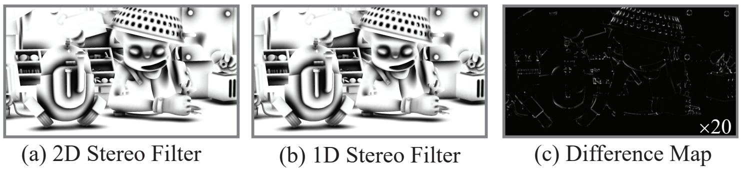

Fig. 7. (a-b): The filtered AO of the left eye using our 2D and 1D stereo filter. (c): The absolute value differences between (a) and (b) are very small as shown by the difference map (differences are magnified 20×). This scene is modified from a Blender demo scene: Junk Shop.

a back-check for $\mathcal{L}$ (Section 3.1). Pixels that fail this check are computed from scratch using our stereo obscurance estimate. It avoids replications and leads to a high-quality output but requires newly-visible pixels to be evaluated separately.

3.3 Stereo-aware bilateral filter

The obscurance estimates from the previous step, when involving a small amount of samples, are noisy and need to be filtered. This is commonly done with a cross-bilateral filter [Eisemann and Durand 2004].

Similar to the obscurance estimate, for a pixel **p**, we combine filter samples from both views to give us the final filtered value $\mathcal{A}(\mathbf{p})$. Specifically, its world space position **P** and the corresponding pixel in the other view, $\mathbf{p}^*$ (the nearest pixel of **p**'s projection) are evaluated in their corresponding image, involving their respective $N \times N$ image space neighborhood:

$$F\left(\mathbf{p}^v\right) = \frac{1}{W\left(\mathbf{p}^v\right)} \sum_{\mathbf{k}_i^v \in \mathbf{N}^v} O\left(\mathbf{k}_i^v\right) G_\sigma\left(\left\|\mathbf{P} - \mathbf{K}_i^v\right\|\right),$$

$$W\left(\mathbf{p}^v\right) = \sum_{\mathbf{k}_i^v \in \mathbf{N}^v} G_\sigma\left(\left\|\mathbf{P} - \mathbf{K}_i^v\right\|\right)$$

The superscript v indicates view-dependent quantities and $\mathbf{p}^v$ equals to **p** or $\mathbf{p}^*$. Hence, $\mathbf{k}_i^v$ is a sample in $\mathbf{p}^v$'s image space neighborhood $\mathbf{N}^v$. $\mathbf{K}_i^v$ is the world-space position of $\mathbf{k}_i^v$. W is a normalization factor and G_σ is a Gaussian function with variance σ:

$$G_\sigma(x) = \frac{e^{-x^2/(2\sigma^2)}}{\sqrt{2\pi}\sigma},$$

where the parameter σ relates to a length in world space; however, the filter's input footprint is defined in image space. Assuming that the world space length r corresponds to an image space

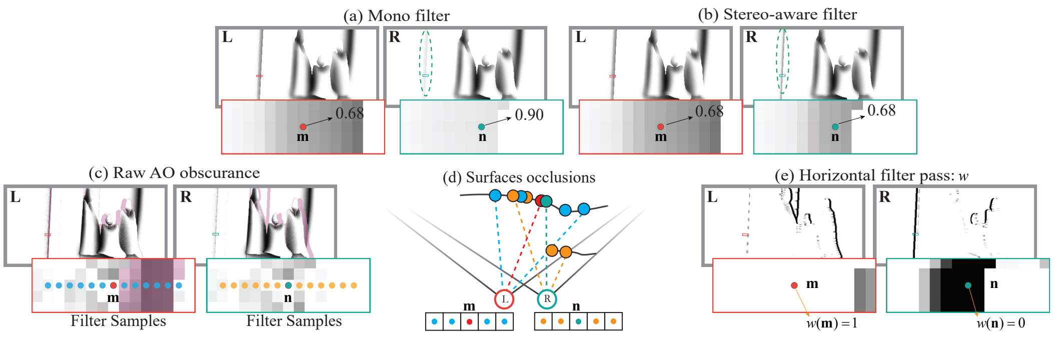

Fig. 8. Our stereo-aware filter reduces inconsistency inherent in the monoscopic filter. The points **m** and **n** indicate two corresponding pixels in each view. (a): Results of the mono filter. (b): The stereo filter produces consistent results in both views. Both filters used the same input (stereo-aware obscurance estimate with RP). (c): Filter samples used by the points in the horizontal pass. Purple pixels mark the pixels that are only visible for one view. Some of the filter samples around **m** are dark, but these are not visible in the other view. (d): Visualization of the situation. (e) In the horizontal filter pass, our weight function w (depicted) determines that both n and m use the filtered value evaluated in the left view. Thus, the results are consistent. The model is from the Sea House scene.

length of N pixels, we set $\sigma = 1/3\, r$. We further clamp σ to the range $[\sigma_{\min}, \sigma_{\max}]$, where the bounds are empirically determined per scene, which is a common approach to avoid overly large kernel sizes.

We blend between the filter values $F(\mathbf{p})$ and $F(\mathbf{p}^*)$ linearly using the weight w:

$$\begin{aligned}\mathcal{A}(\mathbf{p}) &= w\, F(\mathbf{p}) + (1-w)\, F(\mathbf{p}^*)\\ w &= \text{smoothstep}\,(\alpha, 0.5-\eta, 0.5)\\ \alpha &= \frac{W(\mathbf{p})}{W(\mathbf{p}) + W(\mathbf{p}^*)}.\end{aligned}$$

Hereby, we give more weight to the filter samples that are closer to **P**. We rely on the weight $W(\mathbf{p}^v)$ as a measure of the overall distance of the samples in view v to **P**. The variable α describes which view includes closer samples. Figure 6(A-C) shows three examples when the filter samples of **p** and $\mathbf{p}^*$ have similar distance to **P** or different distance to **P**.

The GLSL smoothstep function performs Hermite interpolation [Khronos Group 2014] between zero and one between the edges $0.5 - \eta$ and 0.5, and returns zero and one outside of the edges, respectively. The value η determines the size of the region, where both values affect the result, with $\eta = 0.05$ keeping this region very narrow as shown by Figure 6(D). Any value $\alpha \geq 0.5$ essentially reduces to the monoscopic filter result.

To accelerate, we rely on a separation of the filter, although not theoretically equivalent, it is common practice to apply a bilateral filter in this way in order to reduce filtering costs [Bavoil et al. 2008; Huang et al. 2011; McGuire et al. 2011; NVIDIA 2016; Shanmugam and Arikan 2007]. A filter in the horizontal direction, in which the neighborhood $\mathbf{N}^v$ is reduced to N pixels in a row and their values are combined as described above, is followed by a similar vertical pass, consuming the horizontal pass's output and using a vertical neighborhood of N pixels instead. We found that this leads to a ~ 3 times speedup at a minimal quality loss (PSNR 46.57) for a 13×13 size filter applied to an image at a resolution of 1920 × 1080 (Figure 7). Figure 8 highlights cases that our stereo-aware filter handles more correctly than a monoscopic filtering of each view.

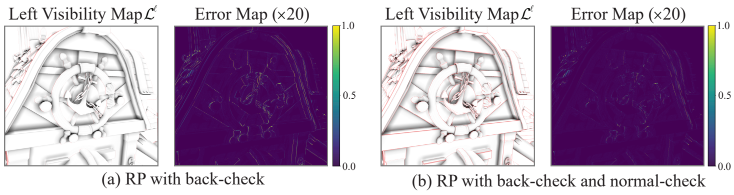
Left Visibility Map $\mathcal{L}^\ell$
Error Map (×20)
1.0
0.5
0.0
(a) RP with back-check
Left Visibility Map $\mathcal{L}^\ell$
Error Map (×20)
1.0
0.5
0.0
(b) RP with back-check and normal-check

Fig. 9. Inconsistency errors for the reference AO. The visibility map is visualized by taking the AO map as background and using purple color to indicate pixels only visible in one eye. These purple pixels are detected by reprojection and can be ignored in the error map. Additional tests on the normal orientation remove ambiguous edge cases. The model is from the Sea House scene.

4 RESULTS AND ANALYSIS

In this section, we cover how inconsistency is evaluated and then explain how each step of our method reduces the inconsistency errors gradually. We will also analyze the rendering cost of our stereo method compared to conventional monoscopic SSAO methods.

4.1 Inconsistency evaluations

Similar to Gong et al.'s way of evaluating stereo inconsistency errors [Gong et al. 2018], we evaluate inconsistency errors by projecting pixels from one view to the other via RP. Let $I(\mathbf{p})$ be the fetched bilinearly-interpolated AO value of $\mathbf{p}$ in the other view. We define the inconsistency error of a pixel as:

$$\mathcal{E}(\mathbf{p}) = |\mathcal{A}(\mathbf{p}) - I(\mathbf{p})|.$$

$\mathcal{E}$ only depicts inconsistency errors for pixels that are visible for both views (see Section 3.1 and Section 3.2 for how visible pixels are classified). As both are similar, we decided to always show the left-view error map ($\mathcal{E}^\ell$).

We generated reference AO values by baking AO maps in Blender. In theory, the inconsistency error map ($\mathcal{E}$) should be zero in this case. However, due to discretization and numerical limitations, some minor inconsistencies remain throughout the image. These are particularly visible at some edges, especially those with a small depth difference but large normal direction changes (Figure 9). These subtle inconsistencies on some surfaces stem from the fact that there is no pixel-wise match between both views; here, we have to compare $\mathbf{p}$'s value to an interpolated value in the other view. In consequence, $\mathcal{E}(\mathbf{p})$ will not be zero. To avoid such numerical issues at the edges, we check whether the normal direction difference between $\mathbf{p}$ and $\mathbf{p}$'s four nearest neighbors in the other view is within a threshold ($\theta = 6°$), which we refer to as *normal-check*. Figure 9 shows the improvement, as some edge regions are correctly classified as only visible to one eye and fewer errors occur. Please also notice that errors are magnified 20 times in Figure 9.

4.2 Step-wise evaluation

As indicated, SSAO algorithms have two parts: the raw obscurance estimation and the bilateral filtering. We can replace each of the two with our stereo version to examine the reduction of inconsistencies.

As shown in Figure 10(A), conventional SSAO methods show large inconsistency errors, while replacing the obscurance estimate with our stereo version, most of the pixels with large errors are corrected (red square in Figure 10(B)). Nevertheless, small errors remain, since the sample sets still differ between the two views (orange square in Figure 10(B)). After introducing RP, the small errors are removed as well because the raw obscurance value is reused (orange square in Figure 10(C)). RP

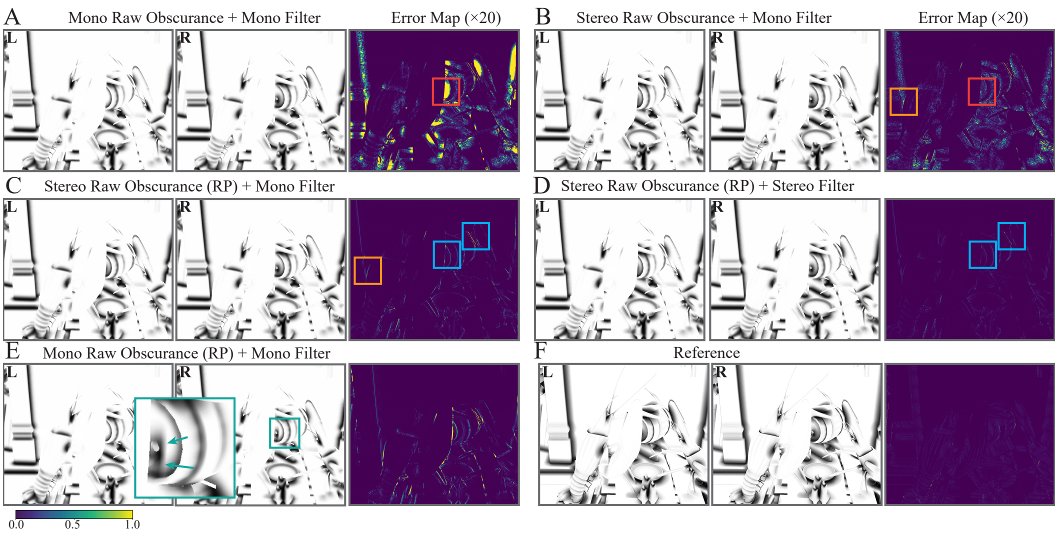
A
Mono Raw Obscurance + Mono Filter
Error Map (×20)
B
Stereo Raw Obscurance + Mono Filter
Error Map (×20)
C
Stereo Raw Obscurance (RP) + Mono Filter
D
Stereo Raw Obscurance (RP) + Stereo Filter
E
Mono Raw Obscurance (RP) + Mono Filter
F
Reference
L
R
0.0
0.5
1.0

Fig. 10. Progressive reductions in inconsistency. A: Plain HBAO+ applied to each eye. B: Just using stereo-aware obscurance estimation, but no RP and with mono filtering. Inconsistencies are already significantly reduced. C: Adding RP further reduces inconsistency (see region highlighted in orange). D: Stereo filtering again reduces the errors (regions highlighted in blue). E: Just reprojecting the mono obscurance results in visible seams. F: Our method reduces inconsistency to a similar level as the geometry-based reference AO solution. Reference is generated by using several baked AO maps (produced in Blender) with resolutions up to 10k×10k. The model is from the Sea House scene.

alone on mono SSAO values fails due to a missing handling of the disocclusions, resulting in clear artifacts (Figure 10(E)). Finally, replacing the mono filter with our stereo one avoids the large errors that the mono filter introduces (Figure 10(D), please compare the blue squares in (C) and (D)). In the end, we achieve inconsistency errors similar to those of the AO reference (Figure 10(F)).

4.3 Performance analysis

We evaluate our method on an Intel i9-11900K 8-core processor (3.5 GHz) with 64 GB RAM and a single NVIDIA GeForce RTX 3090 with 24 GB video memory. We render 3D scenes at 1920 × 1080 pixels (FullHD) for anaglyph images and 2468 × 2740 pixels for HTC Vive Pro (resolution recommended by SteamVR for our desktop). The pixel resolution is given per eye. HBAO+ is computed for both a full and a half-sized image (1/4 pixels). For HBAO+, we use 10 directions and 4 samples per direction. We use a 13-tap size filter kernel. Since screen-space AO performance highly depends on the number of rendered pixels, we make sure that all performance-evaluation frames only contain pixels with an underlying surface on which an AO value is calculated. Tables 2 and 2 show the average time of 400 frames recorded in the Sponza scene [McGuire 2017].

Table 2 compares the rendering time of our stereo raw obscurance estimation with that of mono HBAO+. Table 2(a) shows that the most time-consuming step of generating noisy AO is the raw obscurance estimation. With RP used, the cost of calculating the raw obscurance for the other eye is lowered significantly. In addition, VR headsets have more strict frame rate requirements compared to anaglyph glasses and computing AO at full resolution is too expensive for VR headsets, therefore, computing AO at half resolution is necessary for VR.

Table 1. Render time (in ms): Raw obscurance estimation vs. monoscopic version. $O^{\ell->r}$ is the right-eye AO map after RP. $O^{r,h}$ is the time to evaluate newly-visible pixels. *Full* and *Half* indicate computations taking place on a full and half-sized image, respectively.

(a) Stereo HBAO+ Obscurance Estimation

Resolution Per Eye		$\mathcal{L}^\ell$	$\mathcal{L}^r$	O^ℓ	O^r		Total Time
					$O^{\ell->r}$	$O^{r,h}$	
1920×1080	Full	0.06	0.06	2.36	0.07	0.42	2.97
	Half	0.02	0.02	0.60	0.02	0.15	0.81
2468×2740	Full	0.19	0.19	8.12	0.21	2.01	10.72
	Half	0.05	0.05	2.02	0.06	0.62	2.80

(b) Monoscopic HBAO+ Obscurance Estimation

Resolution Per Eye		O^ℓ	O^r	Total Time
1920×1080	Full	0.98	0.98	1.96
	Half	0.24	0.24	0.48
2468×2740	Full	3.33	3.33	6.66
	Half	0.75	0.75	1.50

Table 2. Render time (in ms): our stereo filter vs. mono filter. X and Y indicate times for the horizontal and vertical passes.

(a) Stereo Filter

Render Resolution	$\mathcal{A}^\ell$		$\mathcal{A}^r$		Total Time
	X	Y	X	Y	
1920×1080	0.32	0.32	0.32	0.32	1.28
2468×2740	1.12	1.11	1.12	1.11	4.46

(b) Monoscopic Filter

Render Resolution	$\mathcal{A}^\ell$		$\mathcal{A}^r$		Total Time
	X	Y	X	Y	
1920×1080	0.12	0.12	0.12	0.12	0.48
2468×2740	0.40	0.40	0.40	0.40	1.60

Compared with the monoscopic raw obscurance estimation in Table 2(b), ours is about $2 \sim 3$ times more expensive due to the fact that sampling the screen-space depth and normal information of two views is required (see the computation time of O^ℓ). Similarly, in Table 2, we can see that our stereo filter is about 3 times as expensive as the mono version for each filter direction both in FullHD and VR-headset resolution. We do not introduce RP into our stereo filter since reprojection of a filtered result may create noticeable seams at pixels that are not visible in the other view. Further performance optimizations of our stereo filter are kept as future work.

5 CONCLUSIONS AND FUTURE WORK

In this paper, we mainly focus on tackling the inconsistency caused by the differences between screen space of the left eye and the right eye. Specifically, we propose a stereo raw obscurance estimation method with RP, which makes sure that the raw obscurance is consistent. With RP included, the rendering time is reduced significantly compared with computing stereo raw obscurance from scratch for one eye. Our proposed stereo-aware filter also reduces the inconsistency in the monoscopic bilateral filter pass.

Our obscurance estimation method samples the depth and normal buffer for two views and the stereo filter pass takes information from both views into account. Both procedures increase the bandwidth requirement significantly and result in higher computation costs compared with mono screen-space AO. However, we believe that the efficiency of our method could be further improved.

While estimating the raw obscurance, we only need more samples of both views for pixels that are fetching from different screen-space geometry information. Therefore, future work can be done to further speed up the computation speed utilizing this property. Importance sampling [Vardis et al. 2013] is a potential solution. At the same time, further efforts can be tried by reusing information from the temporal domain to reduce the number of used samples.

In addition, we also observe that screen space mainly differs in near areas, our method may also be more efficient by only sampling both views for near regions, similar to the idea of hybrid mono-stereo rendering [Fink et al. 2019]. Only giving more samples to the central regions of rendered images in VR since users look at central regions most of the time [Sitzmann et al. 2018] and integrating eye-tracking for evaluating the important pixels by checking where users are looking at on the fly [Shi et al. 2020; Weier et al. 2017] are also promising directions. Verifying these ideas still needs a large amount of work, therefore, we leave improving the efficiency of our method as future work.

Other interesting work may also be further extended based on our contributions: combining our method with Stochastic-Depth AO [Vermeer et al. 2021] to solve the hidden geometry artifacts of screen-space algorithms and keep the stereo consistency at the same time; applying our stereo bilateral filter into ray-tracing denoising for stereo contents; developing other screen-space algorithms into stereo versions, like screen-space stylization [Bléron et al. 2018] and image dithering effect [Velho and Gomes 1995].

ACKNOWLEDGMENTS

We want to thank Dr. L. Scandolo and Dr. M. Stengel for initial discussions and advice on the code. We want to thank Prof. S. Deng, Prof. H. Huang, Prof. Y. Wu and Dr. W. Ma for providing computation resources for P. Shi.

Funding: This work was partially funded by VR-Renovate (403.19.222) - Dutch Research Council (NWO) with consortium partners Noorderberg B.V., Knaapen Renovatie en Onderhoud B.V., Mateboer Bouw B.V. en Zaanderwijk Beheer B.V; the Swiss National Science Foundation (Advanced Postdoc Mobility Project 174321); the China Scholarship Council (No. 201603170298); the National Natural Science Foundation of China (Grant 61871388).

REFERENCES

Tomas Akenine-Möller, Eric Haines, and Naty Hoffman. 2018. *Real-time rendering.* AK Peters/CRC Press.

Louis Bavoil and Johan Andersson. 2012. Stable SSAO in Battlefield 3 with selective temporal filtering. In *Game Developers Conference*, Vol. 12. https://www.gdcvault.com/play/1015538/Stable-SSAO-in-Battlefield-3

Louis Bavoil, Miguel Sainz, and Rouslan Dimitrov. 2008. Image-Space Horizon-Based Ambient Occlusion. In *ACM SIGGRAPH 2008 Talks.* Association for Computing Machinery. https://doi.org/10.1145/1401032.1401061

Alexandre Bléron, Romain Vergne, Thomas Hurtut, and Joëlle Thollot. 2018. Motion-Coherent Stylization with Screen-Space Image Filters. In *Proceedings of the Joint Symposium on Computational Aesthetics and Sketch-Based Interfaces and Modeling and Non-Photorealistic Animation and Rendering (Expressive '18).* Association for Computing Machinery, Article 10. https://doi.org/10.1145/3229147.3229163

Dennis R. Bukenberger, Katharina Schwarz, and Hendrik P. A. Lensch. 2018. Stereo-Consistent Contours in Object Space. *Computer Graphics Forum* 37, 1 (2018), 301–312. https://doi.org/10.1111/cgf.13291

Franklin C. Crow. 1977. Shadow Algorithms for Computer Graphics. In *Proceedings of the 4th Annual Conference on Computer Graphics and Interactive Techniques (SIGGRAPH '77).* Association for Computing Machinery, 242–248. https://doi.org/10.1145/563858.563901

Elmar Eisemann and Frédo Durand. 2004. Flash photography enhancement via intrinsic relighting. *ACM Transactions on Graphics (TOG)* 23, 3 (2004), 673–678. https://doi.org/10.1145/1015706.1015778

Laura Fink, Nora Hensel, Daniela Markov-Vetter, Christoph Weber, Oliver Staadt, and Marc Stamminger. 2019. Hybrid Mono-Stereo Rendering in Virtual Reality. In *2019 IEEE Conference on Virtual Reality and 3D User Interfaces (VR).* 88–96. https://doi.org/10.1109/VR.2019.8798283

Pascal Gautron. 2020. Real-Time Ray-Traced Ambient Occlusion of Complex Scenes Using Spatial Hashing. In *ACM SIGGRAPH 2020 Talks (SIGGRAPH '20).* Association for Computing Machinery, Article 5. https://doi.org/10.1145/3388767.3407375

Xinyu Gong, Haozhi Huang, Lin Ma, Fumin Shen, Wei Liu, and Tong Zhang. 2018. Neural stereoscopic image style transfer. In *Proceedings of the European Conference on Computer Vision (ECCV).* 54–69. https://doi.org/10.1007/978-3-030-01228-1_4

Dejing He, Rui Wang, and Hujun Bao. 2019. Real-Time Rendering of Stereo-Consistent Contours. In *2019 IEEE Conference on Virtual Reality and 3D User Interfaces (VR).* 81–87. https://doi.org/10.1109/VR.2019.8797990

Jing Huang, Tamy Boubekeur, Tobias Ritschel, Matthias Hollaender, and Elmar Eisemann. 2011. Separable Approximation of Ambient Occlusion. In *Short paper at Eurographics*. 1–4. http://graphics.tudelft.nl/Publications-new/2011/HBRHE11

Jorge Jiménez, Xianchun Wu, Angelo Pesce, and Adrian Jarabo. 2016. Practical real-time strategies for accurate indirect occlusion. *SIGGRAPH 2016 Courses: Physically Based Shading in Theory and Practice* (2016).

Khronos Group. 2014. smoothstep - OpenGL 4 Reference Pages. https://www.khronos.org/registry/OpenGL-Refpages/gl4/html/smoothstep.xhtml Accessed: 2021-12-21.

Frank L. Kooi and Alexander Toet. 2004. Visual comfort of binocular and 3D displays. *Displays* 25, 2 (2004), 99–108. https://doi.org/10.1016/j.displa.2004.07.004

Bradford James Loos and Peter-Pike Sloan. 2010. Volumetric Obscurance. In *Proceedings of the 2010 ACM SIGGRAPH Symposium on Interactive 3D Graphics and Games (I3D '10)*. Association for Computing Machinery, 151–156. https://doi.org/10.1145/1730804.1730829

Michael Mara, Morgan McGuire, Derek Nowrouzezahrai, and David Luebke. 2016. Deep G-Buffers for Stable Global Illumination Approximation. In *Eurographics/ ACM SIGGRAPH Symposium on High Performance Graphics*. The Eurographics Association. https://doi.org/10.2312/hpg.20161195

Oliver Mattausch, Daniel Scherzer, and Michael Wimmer. 2010. High-Quality Screen-Space Ambient Occlusion using Temporal Coherence. *Computer Graphics Forum*. https://doi.org/10.1111/j.1467-8659.2010.01784.x

Morgan McGuire. 2010. Ambient Occlusion Volumes *(HPG '10)*. Eurographics Association, 47–56. https://doi.org/10.5555/1921479.1921488

Morgan McGuire. 2017. *Computer Graphics Archive*. https://casual-effects.com/data

Morgan McGuire, Brian Osman, Michael Bukowski, and Padraic Hennessy. 2011. The alchemy screen-space ambient obscurance algorithm. In *Proceedings of the ACM SIGGRAPH Symposium on High Performance Graphics*. 25–32. https://doi.org/10.1145/2018323.2018327

Martin Mittring. 2007. Finding Next Gen: CryEngine 2. In *ACM SIGGRAPH 2007 Courses*. Association for Computing Machinery, 97–121. https://doi.org/10.1145/1281500.1281671

Diego Nehab, Pedro V. Sander, Jason Lawrence, Natalya Tatarchuk, and John R. Isidoro. 2007. Accelerating Real-Time Shading with Reverse Reprojection Caching. In *Proceedings of the 22nd ACM SIGGRAPH/EUROGRAPHICS Symposium on Graphics Hardware (GH '07)*. Eurographics Association, 25–35. https://doi.org/10.5555/1280094.1280098

Lesley Northam, Paul Asente, and Craig S. Kaplan. 2012. Consistent Stylization and Painterly Rendering of Stereoscopic 3D Images. In *Proceedings of the Symposium on Non-Photorealistic Animation and Rendering (NPAR '12)*. Eurographics Association, 47–56. https://doi.org/10.5555/2330147.2330158

NVIDIA. 2016. ShadowWorks. https://developer.nvidia.com/shadowworks Accessed: 2021-10-20.

Perumaal Shanmugam and Okan Arikan. 2007. Hardware Accelerated Ambient Occlusion Techniques on GPUs. In *Proceedings of the 2007 Symposium on Interactive 3D Graphics and Games (I3D '07)*. Association for Computing Machinery, 73–80. https://doi.org/10.1145/1230100.1230113

Peiteng Shi, Markus Billeter, and Elmar Eisemann. 2020. SalientGaze: Saliency-based gaze correction in virtual reality. *Computers & Graphics* 91 (2020), 83–94. https://doi.org/10.1016/j.cag.2020.06.007

Vincent Sitzmann, Ana Serrano, Amy Pavel, Maneesh Agrawala, Diego Gutierrez, Belen Masia, and Gordon Wetzstein. 2018. Saliency in VR: How Do People Explore Virtual Environments? *IEEE Transactions on Visualization and Computer Graphics* 24, 4 (2018), 1633–1642. https://doi.org/10.1109/TVCG.2018.2793599

Kostas Vardis, Georgios Papaioannou, and Athanasios Gaitatzes. 2013. Multi-View Ambient Occlusion with Importance Sampling. In *Proceedings of the ACM SIGGRAPH Symposium on Interactive 3D Graphics and Games (I3D '13)*. Association for Computing Machinery, 111–118. https://doi.org/10.1145/2448196.2448214

Luiz Velho and Jonas Gomes. 1995. Stochastic Screening Dithering with Adaptive Clustering. In *Proceedings of the 22nd Annual Conference on Computer Graphics and Interactive Techniques (SIGGRAPH '95)*. Association for Computing Machinery, 273–276. https://doi.org/10.1145/218380.218452

Jop Vermeer, Leonardo Scandolo, and Elmar Eisemann. 2021. Stochastic-Depth Ambient Occlusion. *Proc. ACM Comput. Graph. Interact. Tech.* 4, 1, Article 3 (April 2021), 15 pages. https://doi.org/10.1145/3451268

Martin Weier, Michael Stengel, Thorsten Roth, Piotr Didyk, Elmar Eisemann, Martin Eisemann, Steve Grogorick, André Hinkenjann, Ernst Kruijff, Marcus Magnor, Karol Myszkowski, and Philipp Slusallek. 2017. Perception-driven Accelerated Rendering. *Computer Graphics Forum* (2017). https://doi.org/10.1111/cgf.13150

Sergey Zhukov, Andrei Iones, and Grigorij Kronin. 1998. An ambient light illumination model. In *Eurographics Workshop on Rendering Techniques*. Springer, 45–55. https://doi.org/10.1007/978-3-7091-6453-2_5