Article Type: Description (see Introduction for more detail)

Efficient Stereo-Aware Screen-Space Ambient Occlusion with Adaptive Computation

Yu-Ting Wu, *National Taipei University, Taiwan*

Abstract—Screen-space ambient occlusion (SSAO) has become a widely used technique in real-time rendering, valued for its high performance and full support for dynamic geometry. However, applying SSAO directly to stereo rendering can result in incorrect depth perception and viewer discomfort due to differences in captured scene approximations between the left and right views. Existing methods for generating stereo-consistent SSAO often involve substantial computational costs. This paper introduces an adaptive method, inspired by Weber's law, to enhance the efficiency of achieving stereo-consistent SSAO. Our method identifies inconsistent pixels generated by cost-effective SSAO algorithms, such as monoscopic SSAO, and selectively applies computationally intensive stereo-aware computations only to those pixels. Experiments demonstrate that our method delivers stereo-consistent results comparable to state-of-the-art techniques while significantly enhancing rendering performance.

Introduction

Ambient occlusion (AO) [4] is a popular technique in computer graphics that enhances scene realism by darkening areas where ambient light is less accessible, such as corners and contact points, thereby improving depth perception. Building on Mittring's work [5], most AO research has been explored in screen space to achieve real-time frame rates and support fully dynamic scenes. These techniques, collectively known as screen-space ambient occlusion (SSAO), involve capturing geometry information, such as depth and surface normals, into texture maps from the camera's perspective. These texture maps are then utilized to approximate the scene geometry and identify occluders around a shading point.

While SSAO methods are widely used for monoscopic (mono) rendering, applying them directly to stereoscopic (stereo) applications, such as virtual reality (VR) and mixed reality (MR), can cause viewer discomfort and disrupt depth perception [6], [7]. This issue arises because the geometry information used for obscurance estimation is view-dependent, resulting in inconsistencies between the left and right views. Furthermore, SSAO methods often rely on post-filtering processes to reduce noise, which can introduce further inconsistences due to sample variations within the filter kernel. To address these issues, Shi et al. [3] proposed a stereo-aware obscurance estimation method and a stereo-aware bilateral filter that leverage screen-space information from both views. While their method significantly improves view consistency, the high computational overhead of stereo-aware processes remains a challenge for applications requiring high frame rates, such as games, VR, and MR.

This paper presents an efficient method for generating stereo-consistent SSAO. Our method is inspired by Weber's law [1], which posits that humans cannot perceive a difference between two pixel values if their intensity difference is below 1%. Walter et al. later empirically adjusted this threshold to 2% in their Lightcuts method for global illumination approximation [8]. Building on this principle, we developed an adaptive technique to reduce unnecessary stereo-aware computations. Our method begins with an initial AO estimation using a computationally efficient setup, such as a mono obscurance estimation method paired with a mono cross-bilateral filter. We then evaluate pixel-wise differences between views to identify pixels with differences exceeding a certain threshold, as these are likely to cause stereo inconsistencies. Stereo-aware algorithms are applied only to these pixels to resolve the inconsistencies.

Digital Object Identifier 10.1109/XXX.0000.0000000

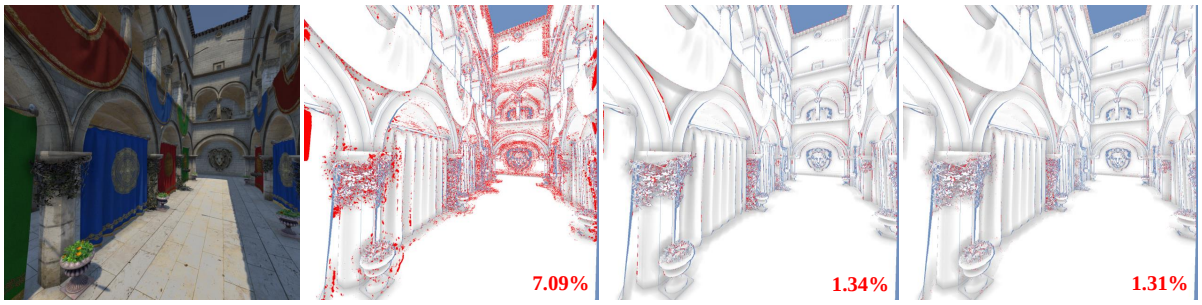

FIGURE 1: **Visualization of the stereo-inconsistent pixels**. The leftmost image displays a view of the SPONZA scene, while the second to fourth images show the ambient occlusion (AO) of that view, generated by various methods using equal samples. Blue pixels represent surfaces visible to only one view. Red pixels highlight stereo-inconsistent areas where differences between views exceed 2%, which could be noticeable to humans according to Weber's law [1]. Applying the monoscopic SSAO method [2] directly in stereo rendering results in numerous inconsistent pixels (7.09%), potentially disrupting depth perception. Shi et al. [3] mitigate these inconsistencies by applying stereo-aware computations to every pixel, though this comes with performance drawbacks. Our method delivers comparable results with significantly improved efficiency. The execution time includes obscurance estimation and cross-bilateral filtering. Due to space constraints, only the right view is shown here.

Experiments show that our method achieves an inconsistency ratio similar to the state-of-the-art stereo-aware SSAO method [3], while significantly enhancing rendering performance. Figure 1 presents an equal-sample comparison of various methods in terms of rendering performance and the number of inconsistent pixels (highlighted in red). Directly applying mono SSAO with filtering in stereo rendering is faster but results in a high inconsistency rate (7.09%). The method proposed by Shi et al. [3] reduces the inconsistency ratio to 1.34%, but it is 2.05× slower than mono SSAO due to the computational cost of stereo-aware processes. Our method limits rendering overhead to 1.28× that of mono SSAO while maintaining a comparable inconsistency ratio. Our key contributions include:

- Our method leverages Weber's law [1] in stereo rendering to identify inconsistent pixels between views, enabling faster stereo-aware SSAO while preserving view consistency.
- Our method offers a balanced approach between rendering speed and stereo consistency, making it particularly beneficial for time-critical applications.

Related Work

Mono ambient occlusion

Ambient occlusion (AO) improves depth perception and scene realism by adjusting the surface color based on the accessibility of a point to ambient light [4]. AO techniques can be categorized into geometry-based or screen-space approaches, depending on how they compute this accessibility. Geometry-based approaches compute occlusion values for a surface point using three-dimensional geometry, often employing ray tracing to identify occluders [9]. To reduce computational costs, some geometry-based methods approximate meshes with proxies, such as oriented disks [10], spheres [11], [12], [13], [14], or voxels [15]. Other methods precompute volumetric obscurance information for each rigid body [16] or triangle [17], which can be accessed by the GPU during runtime. While geometry-based methods are capable of producing high-quality and stereo-consistent AO with complete scene geometry data, their rendering costs are closely tied to scene complexity and are typically too computationally demanding for real-time applications.

Real-time AO approaches are often developed in screen space, where depth maps and normal maps captured from the camera are used to approximate the scene. Obscurance is then estimated by sampling these texture maps through various strategies. Mittring generates 3D random samples within the hemisphere of a surface point [5]. Volumetric obscurance addresses undersampling issues by using line samples [18]. The Horizon-based AO algorithm (HBAO) estimates surface accessibility by calculating the maximum horizon angles from multiple directions [2]. Alchemy ambient obscurance introduces a falloff function to account for the distance between samples and the shading point [17]. HBAO+ combines concepts from both HBAO and Alchemy ambient ob-

scurance [19]. Huang et al. accelerate AO computation using a separable 1-D approximation [20]. Mattausch et al. leverage temporal coherence to reduce the noise and blurring typically seen in conventional SSAO methods [21]. Ritschel et al. extend SSAO to screen-space directional occlusion (SSDO) to account for color shadows and indirect lighting [22].

A common limitation of screen-space methods is the absence of hidden geometry in the geometry buffer, which can lead to underestimates and sudden changes in AO. This issue can be alleviated by using a multi-layer geometry buffer [23] or incorporating multi-view data [24]. Stochastic-depth ambient occlusion (SDAO) enhances the performance of multi-layer approaches by randomly capturing multiple scene layers per pixel using a multi-sampled depth texture [25].

While SSAO methods have been widely used in real-time applications, they are not suitable for stereo applications due to the view-dependent nature of geometry buffers. To overcome this limitation, Shi et al. proposed a stereo-consistent SSAO method that ensures stereo-consistent AO by computing obscurance and performing bilateral filtering using screen-space information from both views [3].

Stereoscopic rendering

Stereoscopic rendering has garnered growing attention with the advent of VR and MR technologies. Several methods for creating stylized stereo images have been introduced, utilizing either traditional techniques [26] or data-driven approaches [7], [27]. Bukenberger et al. [6] and He et al. [28] developed algorithms for rendering stereo-consistent contours. Cheng et al. proposed a spatial-temporal solution for direct volume rendering, employing environment-synced illumination on mixed-reality devices [29]. Nehab et al. introduced a united algorithm called reverse reprojection, which leverages spatial and temporal shading results for stereo and real-time rendering [30].

Several methods have been proposed to reduce rendering costs on VR and MR devices through distance decomposition. A common approach involves replacing distant objects with an environment map [31], [32]. Fink et al. tackled the issue of rendering distant objects twice in both views by utilizing their minimal disparity [33]. Their method aligns with ours in that it applies stereo algorithms only when needed. However, SSAO presents a greater challenge, as it requires considering geometry properties within a local region rather than a single pixel.

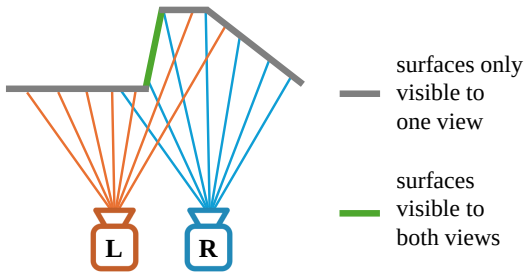

FIGURE 2: **An illustration of the stereo-consistency issue in SSAO**. Occlusion causes variations in the geometry data captured by each view. Surfaces labeled in gray are visible in both views, while those labeled in green are visible in only one. These differences result in inconsistencies during the obscurance estimation and filtering processes in SSAO.

Background

This section starts by highlighting the challenges that stereo-consistent SSAO seeks to resolve. It then presents an overview of the stereo-consistent SSAO approach proposed by Shi et al. [3], which forms the basis of our method.

Problem description

SSAO methods generally sample points around the shading point in world space and project them into the camera view. These methods utilize G-buffers data, such as depth and normal maps, captured from the view to estimate the shading point's obscurance. In stereo applications, computations are required for both the left and right views. A straightforward approach involves applying the SSAO method independently to each view. However, as illustrated in Figure 2, the geometry data captured by each view may differ, as some surfaces are visible only to one view. This discrepancy can lead to inconsistent obscurance estimation and filtering within the SSAO algorithm. To address this challenge, Shi et al. [3] proposed stereo-aware algorithms, which are described in the following subsections.

Stereo-aware obscurance estimation

Figure 3(a) depicts the process of estimating the raw obscurance for a shading point. The method starts by generating samples $\{S_i\}$ within a world-space disk around a shading point P in the left view, following the approach of mono HBAO+ [19]. Each sample S_i is then projected into both the left and right views, where its obscurance contributions are computed using the screen-space data from both views via the HBAO+

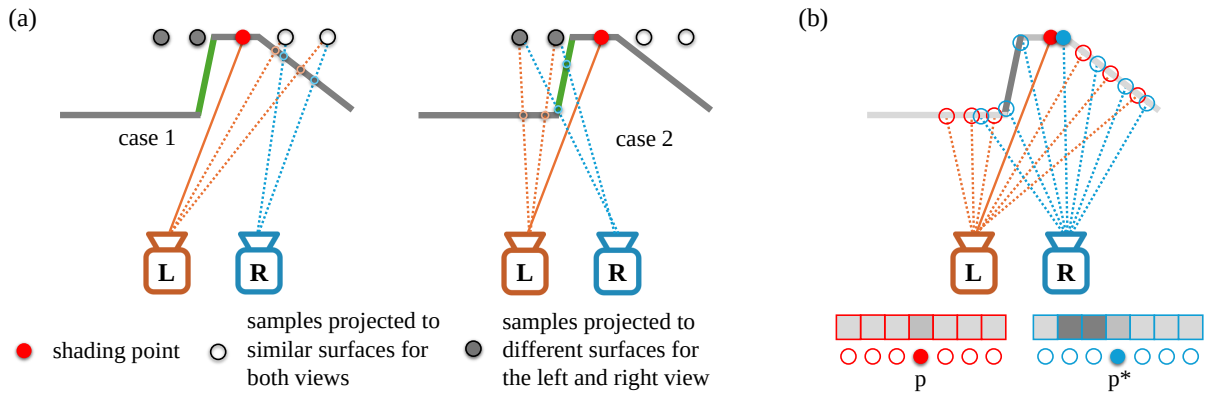

FIGURE 3: **An illustration of (a) stereo-aware obscurance estimation and (b) stereo-aware bilateral filter proposed by Shi et al. [3]**. (a) Occlusion causes variations in the geometry data captured by each view. Surfaces labeled in gray are visible in both views (case 1), while surfaces labeled in green are visible in only one view (case 2). The obscurance value is computed by distinguishing between these two cases and applying different rules to combine the corresponding obscurance values. (b) Filtering samples corresponding to different surfaces can introduce inconsistencies in the filtering processes. Shi et al. address this by combining filtered values from both views, assigning greater weight to the view with higher geometric similarity within the neighborhood. In this scenario, the filtering result of p is assigned greater weight than that of p^*.

algorithm. These two contributions are then combined based on the following rule: if a sample is visible in both views, as indicated by the hollow samples in Figure 3(a), the two obscurance values are averaged; otherwise, as represented by the solid sample, the higher obscurance value is used.

Since stereo-aware obscurance estimation incorporates geometry data from both views, it is performed in the left view, and the results are then reprojected to the right view. To address the oversmoothing caused by naive reprojection [30], Shi et al. [3] enhanced the method by augmenting the reverse reprojection with a back check. This process involves reprojecting the pixel from the right view back to the left view, ensuring sampling results are reused only when the geometry data is confirmed to be consistent.

Stereo-aware bilateral filter

Figure 3(b) illustrates a scenario where independently filtering the raw obscurance for each view may lead to inconsistencies due to filter samples potentially corresponding to different surfaces. To resolve this issue, the stereo-aware bilateral filter combines filtering results from both views to produce the final filtered AO values. For a pixel p and its corresponding pixel p^* in the other view, raw obscurance estimations are smoothed using cross-bilateral filtering [34] within their respective $N \times N$ image-space neighborhoods. This process produces two filtered obscurance values, $F(p)$ and $F(p^*)$. These two filtered values are then blended linearly to derive the final filtered value $A(p)$ for the pixel p:

$$\begin{aligned} A(p) &= wF(p) + (1-w)F(p^*) \\ w &= \text{smoothstep}(\alpha, 0.45, 0.50) \\ \alpha &= W(p)/(W(p) + W(p^*)), \end{aligned} \tag{1}$$

where $W(p)$ and $W(p^*)$ are the cross-bilateral weights to compute $F(p)$ and $F(p^*)$. This approach assigns greater weight to the filtered result with higher geometric similarity within the neighborhood.

While the approach by Shi et al. [3] effectively generates stereo-consistent AO, the two algorithms it employs, stereo-aware obscurance estimation and stereo-aware bilateral filter, introduce considerable computational overheads. This is due to the requirement for simultaneous utilization of screen-space information from both views. As reported in the original paper, this leads to execution times that are approximately 2-3 times slower than those of the conventional monoscopic approach.

Method

The core idea behind our method is to reduce rendering costs by performing computationally expensive stereo-aware operations only on pixels that are likely to exhibit inconsistencies. A straightforward way to identify these pixels is by evaluating the differences in local geometry data between a pixel and its corresponding one in the other view, as SSAO is typically computed based on local geometry. Mattausch et al. employed this strategy to validate temporal reprojection

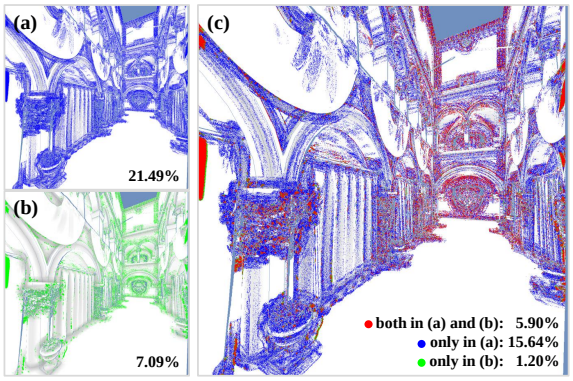

FIGURE 4: **Stereo inconsistency caused by mono obscurance estimation and mono filtering**. (a) and (b) display the inconsistent pixels following the raw obscurance estimation and after the filtering pass, with the inconsistent pixels ratios of 21.49% and 7.09%, respectively. In (c), pixels that are inconsistent in both (a) and (b) are marked in red, while those unique to (a) and (b) are marked in blue and green, respectively. The majority of inconsistent pixels generated by the obscurance estimation are eliminated during the filtering pass. Due to space constraints, only the right view is shown here.

SSAO samples [21]. However, we found that setting a threshold for geometry inconsistency is challenging in practice, as geometry differences do not directly correlate with differences in obscurance values. Additionally, computing geometry differences within the kernel is resource-intensive due to the need for multiple texture fetch operations. For these reasons, we adopted an adaptive approach based on Weber's law.

Weber's law [1] suggests that stereo-consistent AO can be achieved by ensuring the difference between corresponding pixels in each view stays below a certain threshold. Instead of requiring identical AO values for every corresponding pixel pair, it is sufficient to keep their differences within this threshold. Using this principle, we can lower computational costs by performing resource-intensive stereo-aware computations only on pixels where cost-effective setups produce differences that surpass the threshold.

Shi et al. [3] acknowledged that in mono SSAO, both the obscurance estimation pass and the post-filtering pass contribute to inconsistencies between views. However, the extent of inconsistencies introduced by each pass remains unclear. To develop an adaptive method that minimizes the stereo-aware computations, we first analyzed the inconsistencies caused by each pass. As demonstrated in Figure 4, the mono raw obscurance estimation pass generates many inconsistent pixels due to sampling noise. While the mono filtering pass introduces some additional inconsistencies (indicated by the green pixels in (c)), it effectively eliminates most of the inconsistencies generated by the raw obscurance estimation (indicated by the blue pixels in (c)), significantly reducing the overall number of inconsistent pixels.

Building on the preceding analysis, we present the first variant of our adaptive method, as illustrated in the flowchart in Figure 5. Similar to the standard SSAO approach, our method starts by rendering geometry data, including depth and surface normals, into texture maps. Subsequently, pixel-wise correspondences between the left and right views are calculated and stored in dedicated maps. For each pixel, its corresponding texture coordinate in the other view is recorded using the red and green channels.

To identify the pixels that require stereo computation, we first generate a pair of initial AO maps using computationally efficient settings. Our analysis of stereo inconsistencies indicates that applying a mono filtering pass can substantially reduce the number of inconsistent pixels. Therefore, our initial pass employs a mono SSAO method [2] followed by a mono cross-bilateral filter [34]. We then compute the relative differences between the corresponding pixels in the initial AO maps. Pixels are labeled as inconsistent in the adaptive maps if their differences exceed a specified threshold T:

$$\frac{|A(p) - A(p^*)|}{A(p)} \geq T. \tag{2}$$

Here, p and p^* represent a pair of corresponding pixels between the left and right views. $A(p)$ and $A(p^*)$ are the filtered AO values stored in the initial AO maps. Following the recommendation from Lightcuts [8], we set the threshold T to 0.02 (2%).

The adaptive maps guide the subsequent stereo-aware computations. For pixels identified as inconsistent in the left view, we perform stereo-aware obscurance estimation [3], computing the obscurance value using the screen-space data from both views. These computed obscurance values are then reprojected to the right view. Finally, we apply stereo-aware cross-bilateral filtering [3] to reduce the noise in the newly computed pixels. For pixels labeled as consistent in the adaptive maps, we directly copy the filtered results from the initial maps to minimize computation.

While the above method, referred to as Variant 1, generates stereo-consistent results, we discovered that using a mono obscurance estimation method to generate the initial AO maps can sometimes lead to incorrect final results. This issue arises because the mono obscurance estimation may produce incorrect

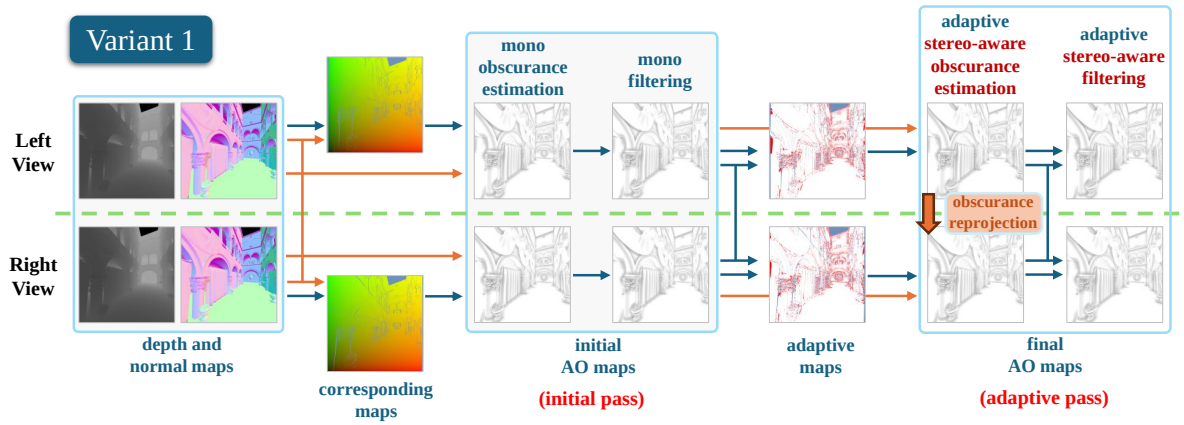

FIGURE 5: **Flowchart of our method (Variant 1)**. Similar to mono SSAO methods, our approach starts by rendering geometry data, including depth and normals, into texture maps for both the left and right views. We then compute the correspondence between views using reprojection with a back check. The initial AO maps are generated using a cost-efficient mono obscurance estimation and mono cross-bilateral filtering, followed by pixel-wise consistency checks. Adaptive maps are subsequently created to identify pixels where differences exceed a predefined threshold. Finally, stereo-aware computations are applied to correct these inconsistent pixels, guided by the adaptive maps.

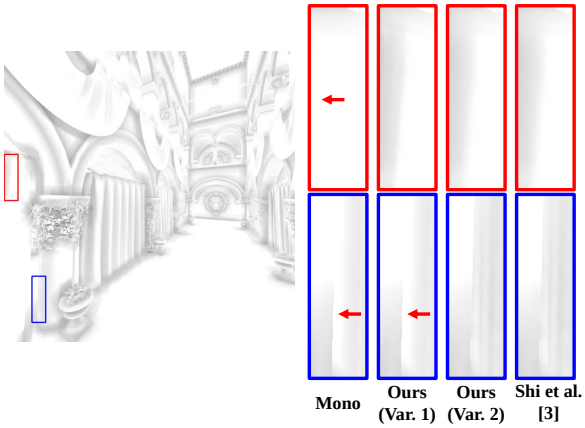

FIGURE 6: **Visual comparison of the two variants of our method**. Using the mono SSAO algorithm to generate the initial AO maps may result in incorrect obscurance estimation due to missing geometry in the G-Buffers. In Variant 1, the stereo-aware obscurance estimation can only correct this issue if the differences between views exceed the threshold defined by Weber's law (red inset). Otherwise, the pixels will be deemed consistent, even if their estimations are inaccurate (blue inset). In contrast, Variant 2 avoids this issue by employing stereo-aware obscurance estimation rather than the mono approach during the initial pass, resulting in outcomes that more closely align with the method proposed by Shi et al. [3].

AO values due to hidden geometry in the G-buffers. As shown in Figure 6, the mono method can underestimate (red and blue insets) the AO value. If the difference between the obscurance values estimated for the left and right views exceeds the threshold defined by Weber's law, the adaptive pass applies stereo-aware obscurance estimation to correct the AO values, as shown in the red inset. However, when both views produce incorrect yet similar estimations, the pixel is classified as consistent in the adaptive map. Because the values appear consistent despite being inaccurate, Variant 1's adaptive pass does not apply stereo-aware computation to these pixels, resulting in incorrect AO values persisting in the final output, as highlighted in the blue inset.

To address the previously mentioned issue, we introduced an alternative approach referred to as Variant 2. As illustrated in Figure 7, Variant 2 retains most stages of Variant 1 but replaces the mono raw obscurance estimation for the initial AO maps with stereo-aware obscurance estimation [3]. This adjustment eliminates the need for adaptive stereo-aware obscurance estimation in the adaptive pass. By incorporating stereo-aware obscurance estimation which utilizes G-buffers data from both views, Variant 2 produces results that more closely align with those of Shi et al. [3] and geometry-based approaches. For the comparisons of AO quality, please refer to Figure 9, and see Table 3 for performance profiling.

Our experiments show that Variant 2 runs slower than Variant 1 due to the application of stereo-aware obscurance estimation to every pixel during the initial pass. As indicated by our performance profiling in

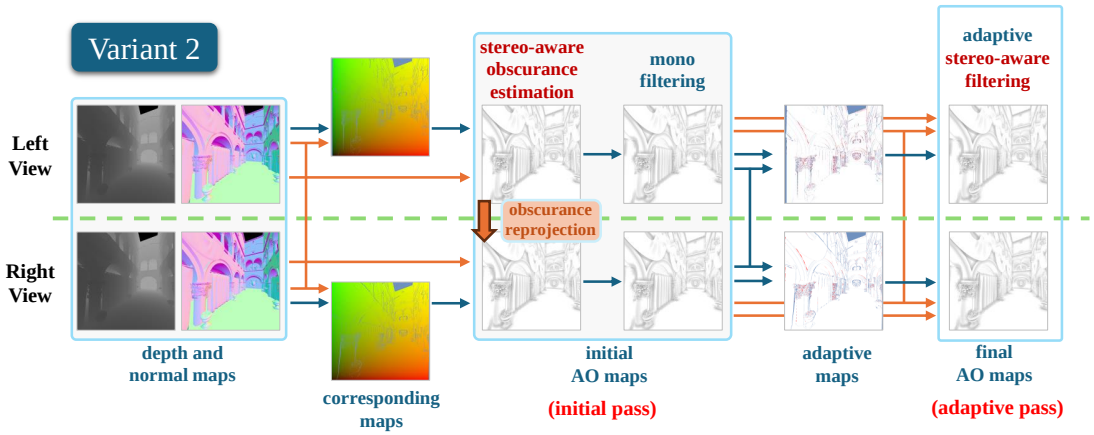

FIGURE 7: **Flowchart of our method (Variant 2)**. Variant 2 differs from Variant 1 by shifting stereo-aware obscurance estimation from the adaptive pass to the initial pass. While this enhances the accuracy of obscurance estimation, it also increases computation time, as the resource-intensive stereo-aware estimation is applied to every pixel.

Table 3, stereo-aware obscurance estimation requires more than twice the time of its mono counterpart. Although Variant 2 reduces computation by omitting adaptive stereo-aware obscurance estimation in the adaptive pass, these savings do not fully compensate for the increased initial cost.

Experiments

Our method was implemented in the Unity engine [35]. All images in this paper were rendered at a resolution of 1254×1254 pixels, corresponding to half the per-eye resolution of the HTC VIVE XR Elite. Performance evaluations were conducted on a desktop equipped with an Intel i7-10700 CPU at 2.90 GHz, 72GB of RAM, and an NVIDIA RTX A4000 graphics card. Following the approach by Shi et al. [3], we employed HBAO+ [19] for raw obscurance estimation and a cross-bilateral filter [34] for noise reduction. To optimize performance during the filtering processs, we split it into horizontal and vertical passes [2], [19], [3], [20], [36]. We use world-space coordinates as geometric features and set the spatial standard deviation to 3.0.

Comparisons

We compared our method with the following settings:

- **Mono**: apply conventional mono HBAO+ [19] and cross-bilateral filtering [34] to both views independently.
- **FullyStereoAware (FSA)**: perform stereo-aware obscurance estimation and stereo-aware bilateral filtering [3] for every pixel.

We assessed all methods using the following test scenes: MARKET, SPONZA, SIBENIK, and BISTRO. Figure 8 highlights the inconsistent pixels produced by different methods from a given view. Pixel consistency was assessed using Eq. 2, with the percentage of inconsistent pixels and execution time for obscurance estimation and filtering shown below each image.

Across the test scenes, using a mono SSAO method combined with a mono bilateral filter (referred to as the **Mono** configuration) achieved the shortest execution time but resulted in a high number of inconsistent pixels. For instance, even in a simple scene like MARKET, this configuration produced 2.47% inconsistent pixels. The **FSA** method, which applies stereo-aware obscurance estimation and stereo-aware cross-bilateral filtering to all pixels, eliminates most inconsistencies but takes more than twice the time of the mono approach. Both variants of our methods achieve stereo consistency comparable to **FSA** while significantly enhancing rendering performance. Variant 1, which utilizes mono obscurance estimation, is faster but exhibits a slightly higher inconsistent rate compared to **FSA**. Conversely, Variant 2, which uses stereo-aware obscurance estimation, takes longer but delivers better stereo consistency.

From the test scenes, we observed that inconsistencies are more prevalent in areas with higher occlusion values (darker pixels in the AO maps). This phenomenon can be attributed to two factors. First, local geometry tends to be more complex near object corners or in regions with contact shadows, resulting in greater sampling variance and more pronounced geometry differences between views within the neighbor-

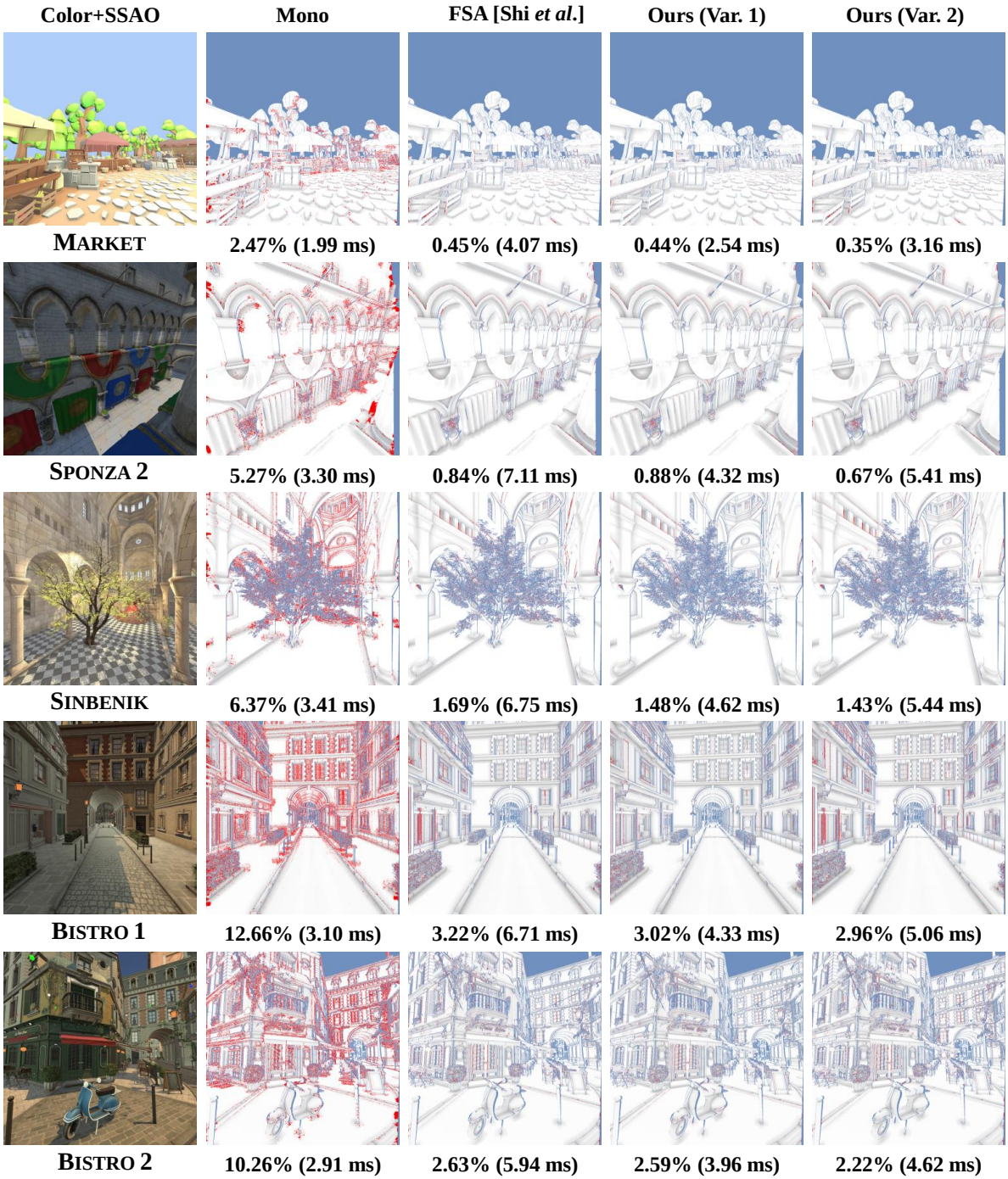

FIGURE 8: **Visualization of inconsistent pixels (highlighted in red)**. The scenes, arranged from top to bottom, are MARKET, SPONZA (view2), SIBENIK, BISTRO (view1), and BISTRO (view2). These examples demonstrate that the **Mono** configuration, which uses mono obscurance estimation and filtering, achieves the shortest execution time but generates the highest number of inconsistent pixels. While **FSA** effectively reduces the inconsistency ratio, it encounters performance challenges, requiring more than twice the execution time of the **Mono** configuration. Our Variant 1 and Variant 2 deliver stereo consistency comparable to **FSA** while substantially improving rendering efficiency. Blue pixels represent skybox areas or regions lacking correspondence between views. The execution time accounts for both obscurance estimation and cross-bilateral filtering. For clarity and due to space limitations, only the right view is shown.

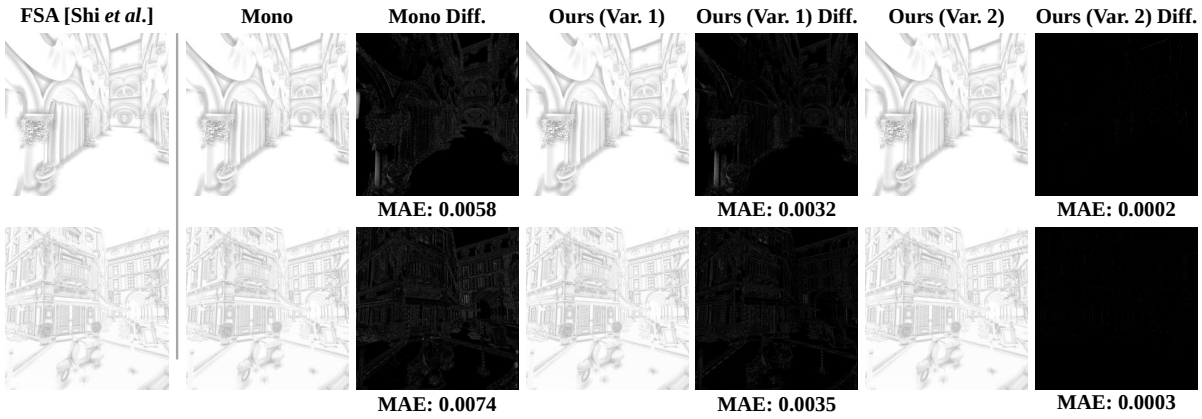

FIGURE 9: **Comparisons of AO quality**. We compute the mean absolute error (MAE) for **Mono** and our two variants, using **FSA** as reference. Variant 1 reduces errors compared to **Mono** by applying stereo obscurance and filtering to certain inconsistent pixels. Variant 2 achieves results closely resembling those of **FSA**, as it utilizes stereo obscurance estimation across all pixels.

	Mono	FSA	Ours (Var.1)	Ours (Var.2)
Sponza	5.82%	1.13%	1.07%	0.90%
Bistro	7.29%	1.99%	1.92%	1.76%

TABLE 1: Stereo inconsistency ratios averaged along a path for **Mono**, **FSA**, and our proposed methods.

	Mono	Ours (Var.1)	Ours (Var.2)
Sponza	0.0049	0.0025	0.0002
Bistro	0.0057	0.0025	0.0002

TABLE 2: Mean absolute error (MAE) averaged along a path for **Mono** and our proposed methods, with **FSA** results as the reference.

hood. Second, according to Weber's law, differences in darker pixels are more noticeable to viewers compared to brighter ones, even when the intensity difference remains the same.

We evaluated the AO quality of SPONZA (Figure 1) and BISTRO 2 (Figure 8) using various methods. In this experiment, results generated by **FSA** were used as reference images, as **FSA** has been shown to closely approximate ray-traced solutions compared to traditional SSAO techniques. These images also represent the highest quality achievable with our method since **FSA** performs stereo-aware computations at every pixel. Figure 9 displays the AO maps generated by the **Mono** configuration and our two variants, along with their differences relative to **FSA**. While Variant 1 improves upon **Mono** by correcting some pixels, Variant 2 produces results nearly identical to **FSA**.

For the SPONZA and BISTRO scenes, we examined the progression of inconsistency ratio and mean absolute error (MAE) along a predefined path. As shown in the progression diagram (Figure 10), both Variant 1 and Variant 2 of our method achieve inconsistency ratios comparable to **FSA**, with Variant 2 also delivering AO quality similar to **FSA** along the path. Table 1 and 2 summarize the average inconsistency ratios and errors for different methods across the two scenes, respecively.

Performance profiling

Table 3 presents detailed performance profiles for the MARKET and SPONZA scene. In the **Mono** and **FSA** configurations, the SSAO computation time is measured during the **initial pass**. Due to the high computational demands of stereo-aware obscurance estimation and bilateral filtering, **FSA** requires more than twice the execution time of the **Mono** configuration.

Our methods include both initial and adaptive passes. In Variant 1, the **Mono** configuration is used for the initial pass, followed by adaptively processing of a small subset of pixels (ranging from 2.47% to 12.66%, as indicated by the inconsistency ratio of the **Mono** configuration in Figure 1 and 8) to achieve stereo consistency. Compared to the **Mono** configuration, the additional overhead ranges from approximately 25% to 40% in our test scenes. These overheads are significantly lower than those of **FSA** and are roughly proportional to the number of inconsistent pixels generated by the **Mono** method in the initial pass.

Variant 2 employs the stereo-aware obscurance estimation proposed by Shi et al. [3], followed by a mono cross-bilateral filter [34] in the initial pass. While this approach increases computation time by applying the resource-intensive stereo-aware obscurance estimation to every pixel, it reduces the number of

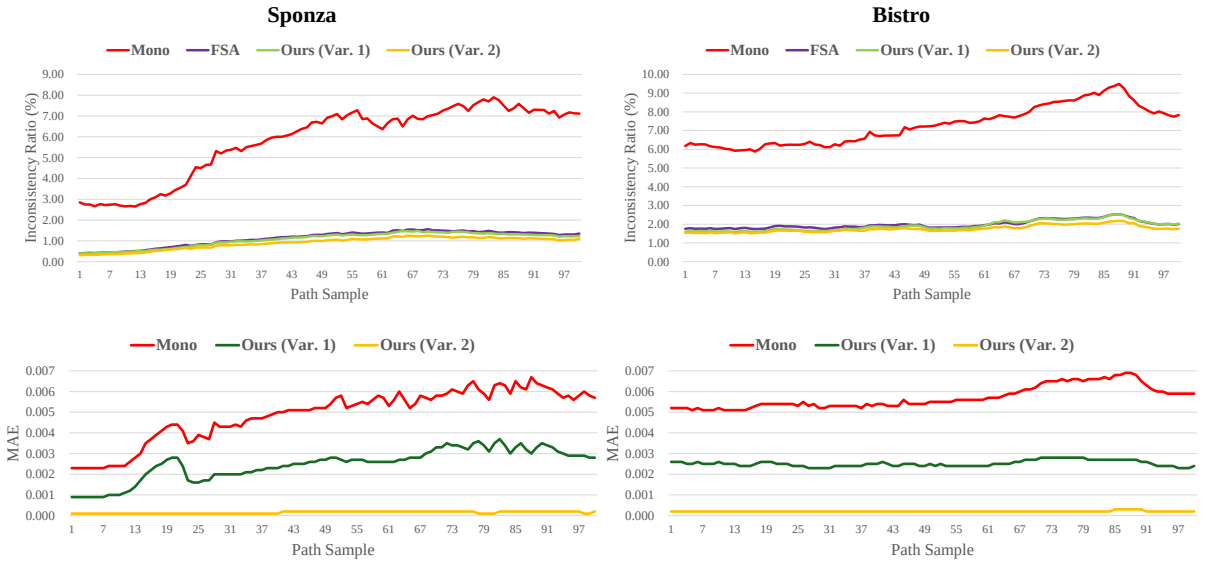

FIGURE 10: **Comparison of stereo inconsistency and AO quality along a predefined path.** The evaluation is conducted using 100 camera poses uniformly sampled along a path in the SPONZA and BISTRO scenes. The top row presents the inconsistency ratios produced by different methods, while the bottom row displays the mean absolute error (MAE) for **Mono** and our two variants, using **FSA** as reference.

inconsistent pixels in the initial AO maps, leading to a faster adaptive pass. Furthermore, Variant 2 eliminates the need for additional obscurance estimation during the adaptive pass. Overall, Variant 2 is approximately 15% to 25% slower than Variant 1 but delivers more accurate AO estimation (as shown in Figure 6 and 9) and a slight improvement in the inconsistent ratio. Users can select between the two variants based on their computational resources and performance requirements.

Stereo consistency v.s. performance

Another advantage of our method is its ability to balance stereo consistency with performance. By adjusting the threshold based on Weber's law, we can manage the execution time. Figure 11 illustrates the execution time of our methods in relation to the inconsistency threshold, tested on SPONZA using the view shown in Figure 1. A stricter (smaller) Weber's law threshold results in more inconsistent pixels, which necessitates additional time during the adaptive pass to address these inconsistencies.

Figure 11 reveals an interesting observation: the execution time for Variant 1 and Variant 2 becomes relatively similar when the inconsistency threshold is set 0.01. This is because the strict threshold results in a large number of inconsistent pixels in the initial AO maps for Variant 1, requiring an extended adaptive pass to resolve these inconsistencies. However,

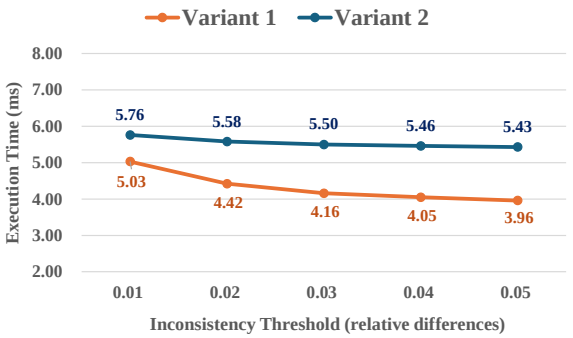

FIGURE 11: **Inconsistency threshold v.s. execution time**. Using a higher threshold permits greater pixel differences between views, which reduces the amount of stereo-aware computations and shortens the execution time. This experiment was conducted on SPONZA, using the view depicted in Figure 1.

with a higher threshold, Variant 1 consistently outpaces Variant 2 in speed, as many computationally expensive stereo-aware obscurance estimations are replaced with more cost-effective mono estimations. Figure 12 visualizes the inconsistent pixels in the initial AO maps generated by Variant 1 and Variant 2 using various thresholds.

Limitations

Similar to **FSA**, our methods cannot completely eliminate stereo inconsistency due to discretization and

Scene	Method	Initial Pass		Adap. map	Adaptive Pass		Total Time
		Obscur. Est.	Filtering		Obscur. Est.	Filtering	
MARKET	Mono	0.85	1.14				1.99
	FSA	1.81	2.26				4.07
	Ours (Var. 1)	0.85	1.14	0.12	0.18	0.25	2.54
	Ours (Var. 2)	1.81	1.14	0.12		0.09	3.16
SPONZA (Fig.1)	Mono	1.38	2.06				3.44
	FSA	3.22	3.82				7.04
	Ours (Var. 1)	1.38	2.06	0.19	0.33	0.46	4.42
	Ours (Var. 2)	3.22	2.06	0.19		0.11	5.58

TABLE 3: **Performance profiling of various configurations**. Both the initial and the adaptive passes involve two processes: raw obscurance estimation (Obscur. Est.) and bilateral filtering (Filtering). Our methods also include the computation of adaptive maps to identify inconsistent pixels in the AO maps generated during the initial pass. All measurements are presented in milliseconds.

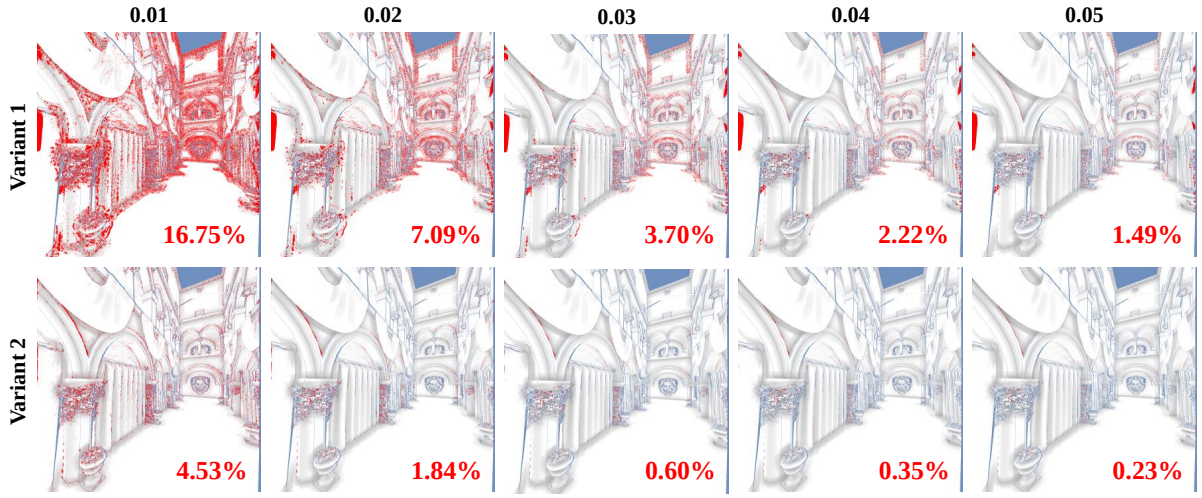

FIGURE 12: Visualization of inconsistent pixels (highlighted in red) in the initial AO maps for various inconsistency thresholds. Due to space limitations, only the right view is displayed here.

numerical limitations. Additionally, the stereo-aware bilateral filter used in both **FSA** and our methods does not ensure identical filtered AO values between views, as the α computed from the bilateral weights in Eq. 1 may differ.

The efficiency of our method relative to **Mono** and **FSA** depends on the level of inconsistency in the initial AO maps. Specifically, it is less efficient than **Mono** when inconsistencies are minimal and less efficient than **FSA** when inconsistencies are widespread. However, in most scenes, our method achieves significant performance improvements, as evidenced by the test cases.

Conclusion

This paper focuses on efficiently computing stereo-consistent SSAO. To address the performance challenges associated with stereo-aware computations, we incorporated Weber's law and developed a method that minimizes unnecessary computations that would go unnoticed by viewers. Our method is adaptive: it begins by generating initial AO maps for each view using relatively inexpensive SSAO algorithms. We then identify inconsistent pixels between views based on Weber's law. Finally, we apply stereo-aware obscurance estimation and cross-bilateral filtering only to those pixels to resolve inconsistencies. This approach significantly reduces the amount of costly stereo-aware computations. Experiments demonstrate that, compared to the state-of-the-art stereo-consistent SSAO method, our method provides substantially improved performance while maintaining comparable consistency. These performance enhancements are particularly advantageous for real-time stereo applications, such as VR and MR.

In the future, we plan to extend our method to other screen-space rendering algorithms, such as screen-space directional occlusion (SSDO) [22]. Ensuring stereo consistency in SSDO presents greater chal-

lenges compared to SSAO, as the sampling process for indirect lighting computation is generally more susceptible to noise.

Another promising research direction is the the integration of temporal reprojection [30], [21] to exploit temporal coherence. By reducing noise in AO, temporal reprojection allows for a lower sample count in obscurance estimation or a smaller filter size by reprojecting previous AO results onto the current frame. When implementing this approach, it is preferable to leverage temporal coherence during the initial pass, as the final results must maintain stereo consistency between views.

Acknowledgments

We thank the anonymous reviewers for their valuable comments and the creators or providers of the models and textures used in this paper, downloaded from McGuire Computer Graphics Archive [37], Open Research Content Archive (ORCA) [38], and unity asset store. This work was supported in part by the National Science and Technology Council (NSTC) under grants 111-2222-E-305-001-MY2.

References

1. H. R. Blackwell, “Luminance difference thresholds,” in *Visual Psychophysics*, D. Jameson and L. M. Hurvich, Eds. Springer Berlin Heidelberg, 1972, pp. 78–101.
2. L. Bavoil, M. Sainz, and R. Dimitrov, “Image-space horizon-based ambient occlusion,” in *ACM SIGGRAPH 2008 Talks*, 2008.
3. P. Shi, M. Billeter, and E. Eisemann, “Stereo-consistent screen-space ambient occlusion,” *ACM Comput. Graph. Interact. Tech.*, vol. 5, no. 1, may 2022.
4. S. Zhukov, A. Iones, and G. Kronin, “An ambient light illumination model,” in *Rendering Techniques*, 1998, pp. 45–55.
5. M. Mittring, “Finding next gen: Cryengine 2,” in *ACM SIGGRAPH 2007 Courses*, 2007, p. 97–121.
6. D. R. Bukenberger, K. Schwarz, and H. P. A. Lensch, “Stereo-consistent contours in object space,” *Computer Graphics Forum*, vol. 37, no. 1, pp. 301–312, 2018.
7. X. Gong, H. Huang, L. Ma, F. Shen, W. Liu, and T. Zhang, “Neural stereoscopic image style transfer,” in *Proc. European Conference on Computer Vision (ECCV)*, 2018, pp. 56–71.
8. B. Walter, S. Fernandez, A. Arbree, K. Bala, M. Donikian, and D. P. Greenberg, “Lightcuts: a scalable approach to illumination,” *ACM Trans. Graph. (Proc. SIGGRAPH)*, vol. 24, no. 3, p. 1098–1107, jul 2005.
9. P. Gautron, “Real-time ray-traced ambient occlusion of complex scenes using spatial hashing,” in *ACM SIGGRAPH 2020 Talks*, 2020.
10. M. Bunnell, “Dynamic ambient occlusion and indirect lighting,” in *GPU Gem2*. NVidia Corporation, 2005, ch. 14, pp. 223–234.
11. K. Hegeman, S. Premože, M. Ashikhmin, and G. Drettakis, “Approximate ambient occlusion for trees,” in *Proc. Symposium on Interactive 3D Graphics and Games*, 2006, p. 87–92.
12. Z. Ren, R. Wang, J. Snyder, K. Zhou, X. Liu, B. Sun, P.-P. Sloan, H. Bao, Q. Peng, and B. Guo, “Real-time soft shadows in dynamic scenes using spherical harmonic exponentiation,” in *ACM Trans. Graph. (Proc. SIGGRAPH)*, 2006, p. 977–986.
13. P. Shanmugam and O. Arikan, “Hardware accelerated ambient occlusion techniques on gpus,” in *Proc. Symposium on Interactive 3D Graphics and Games*, 2007, p. 73–80.
14. P.-P. Sloan, N. K. Govindaraju, D. Nowrouzezahrai, and J. Snyder, “Image-based proxy accumulation for real-time soft global illumination,” in *Proc. Pacific Conference on Computer Graphics and Applications*, 2007, p. 97–105.
15. C. K. Reinbothe, T. Boubekeur, and M. Alexa, “Hybrid ambient occlusion,” in *Eurographics 2009 Areas Papers*, 2009.
16. J. Kontkanen and S. Laine, “Ambient occlusion fields,” in *Proc. Symposium on Interactive 3D Graphics and Games*, 2005, p. 41–48.
17. M. McGuire, “Ambient occlusion volumes,” in *Proc. High-Performance Graphics*, June 2010.
18. B. J. Loos and P.-P. Sloan, “Volumetric obscurance,” in *Proc. Symposium on Interactive 3D Graphics and Games*, 2010, p. 151–156.
19. “Horizon-based ambient occlusion plus (HBAO+),” accessed on June 1, 2024. [Online]. Available: https://developer.nvidia.com/rendering-technologies/horizon-based-ambient-occlusion-plus
20. J. Huang, T. Boubekeur, T. Ritschel, M. Holländer, and E. Eisemann, “Separable approximation of ambient occlusion,” in *Eurographics 2011 Short papers*, 2011.
21. O. Mattausch, D. Scherzer, and M. Wimmer, “High-quality screen-space ambient occlusion using temporal coherence,” *Computer Graphics Forum*, vol. 29, no. 8, pp. 2492–2503, 2010.
22. T. Ritschel, T. Grosch, and H.-P. Seidel, “Approximating dynamic global illumination in image space,”

in *Proc. Symposium on Interactive 3D Graphics and Games*, 2009, p. 75–82.

23. M. Mara, M. McGuire, D. Nowrouzezahrai, and D. Luebke, "Deep G-Buffers for stable global illumination approximation," in *Proc. High-Performance Graphics*, 2016.
24. K. Vardis, G. Papaioannou, and A. Gaitatzes, "Multi-view ambient occlusion with importance sampling," in *Proc. Symposium on Interactive 3D Graphics and Games*, 2013, p. 111–118.
25. J. Vermeer, L. Scandolo, and E. Eisemann, "Stochastic-depth ambient occlusion," *ACM Comput. Graph. Interact. Tech.*, vol. 4, no. 1, apr 2021.
26. L. Northam, P. Asente, and C. S. Kaplan, "Consistent stylization and painterly rendering of stereoscopic 3d images," in *Proc. Symposium on Non-Photorealistic Animation and Rendering*, 2012, p. 47–56.
27. D. Chen, L. Yuan, J. Liao, N. Yu, and G. Hua, "Stereoscopic neural style transfer," in *Proc. IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2018, pp. 6654–6663.
28. D. He, R. Wang, and H. Bao, "Real-time rendering of stereo-consistent contours," in *Proc. IEEE Conference on Virtual Reality and 3D User Interfaces (VR)*, 2019, pp. 81–87.
29. H. Cheng, C. Xu, X. Chen, Z. Chen, J. Wang, and L. Zhao, "Realistic volume rendering with environment-synced illumination in mixed reality," in *Proc. IEEE International Symposium on Mixed and Augmented Reality Adjunct (ISMAR-Adjunct)*, oct 2023, pp. 423–428.
30. D. Nehab, P. V. Sander, J. Lawrence, N. Tatarchuk, and J. R. Isidoro, "Accelerating real-time shading with reverse reprojection caching," in *Proc. ACM SIGGRAPH/EUROGRAPHICS Symposium on Graphics Hardware*, 2007, p. 25–35.
31. V. Popescu, S. H. Lee, A. S. Choi, and S. Fahmy, "Complex virtual environments on thin vr systems through continuous near-far partitioning," in *Proc. IEEE International Symposium on Mixed and Augmented Reality (ISMAR)*, 2022, pp. 35–43.
32. Y. Zhou and V. Popescu, "CloVR: Fast-startup low-latency cloud VR," *IEEE Trans. on Visualization and Computer Graphics*, vol. 30, no. 05, pp. 2337–2346, may 2024.
33. L. Fink, N. Hensel, D. Markov-Vetter, C. Weber, O. Staadt, and M. Stamminger, "Hybrid mono-stereo rendering in virtual reality," in *Proc. IEEE Conference on Virtual Reality and 3D User Interfaces (VR)*, 2019, pp. 88–96.
34. E. Eisemann and F. Durand, "Flash photography enhancement via intrinsic relighting," *ACM Trans. Graph. (Proc. SIGGRAPH)*, vol. 23, no. 3, p. 673–678, aug 2004.
35. Unity Technologies, "Unity," 2023, game development platform. [Online]. Available: https://unity.com/
36. M. McGuire, B. Osman, M. Bukowski, and P. Hennessy, "The alchemy screen-space ambient obscurance algorithm," in *Proc. High-Performance Graphics*, 2011, p. 25–32.
37. M. McGuire. (2017, July) Computer graphics archive. [Online]. Available: https://casual-effects.com/data
38. A. Lumberyard, "Amazon lumberyard bistro, open research content archive (orca)," July 2017. [Online]. Available: http://developer.nvidia.com/orca/amazon-lumberyard-bistro

Yu-Ting Wu is an assistant professor in the Department of Computer Science and Information Engineering at National Taipei University. His research interests encompass computer graphics, extended reality, computer vision, and visual effects. He received his B.S. and M.S. from National Chiao Tung University in 2007 and 2009 respectively, and his Ph.D. from National Taiwan University in 2014, all in Computer Science. Please contact him at yutingwu@mail.ntpu.edu.tw.