Screen-Space Ambient Occlusion for Light Field Displays

Oleksii Doronin
Holografika
Budapest, Hungary
o.doronin@holografika.com

Peter A. Kara
Kingston University
London, UK
p.kara@kingston.ac.uk

Attila Barsi
Holografika
Budapest, Hungary
a.barsi@holografika.com

Maria G. Martini
Kingston University
London, UK
m.martini@kingston.ac.uk

ABSTRACT

In this paper, we discuss the challenge of generating the screen-space ambient occlusion (SSAO) visual effect on the state-of-the-art HoloVizio light field display. We detail the main features of modern SSAO techniques that are currently being applied during visualization on conventional 2D displays, and describe difficulties that potentially can appear when implementing this visual effect on 3D light field or similar systems. The main contribution of this paper is our own modification of the SSAO algorithm for light field displays. The paper also includes suggestions on the possible ways to improve its visual quality and performance.

Keywords

Screen-space ambient occlusion (ssao), surface shading, light field, light field display, algorithm performance.

1 INTRODUCTION

Ambient occlusion (AO) is a technique of modeling the behavior of ambient light. It takes into account all irregularities of the surface of a virtual scene. After this technique is applied, rough surfaces look more contrast and realistic. This effect creates an impression that the scene elements with relatively bigger depth appear darker, while elements on the top remain light (e.g., like at the pencil drawing of a human face on Figure 1). Physical explanation of the described phenomenon is that the incoming light is easily reflected by the top parts of the surfaces, but vanishes at their bottoms because of multiple reflections with absorption.

Currently the main approaches to implement AO are the global illumination techniques [Chris10, Dim08, Tab04], the methods using simplified scene geometry [Bun05, Laine10, Shan07], and the *screen-space ambient occlusion* (SSAO) techniques [Hoang10, Mitt07].

The global illumination approach is based on the simulation of the physical properties of light itself. It is usually implemented in graphical frameworks that target the generation of complex photo-realistic effects. This includes caustics, multiple reflections, refractions, atmospheric conditions, indoor or outdoor lighting, etc. The ambient occlusion effect is present here as a natural consequence of the physical behavior of light. The main drawback of this approach is that it is usually far away from being able to run in real-time, which results in its applications for offline-only systems, like the ones that are used in the production of photo-realistic posters, movies, or design solutions (e.g., architectural design).

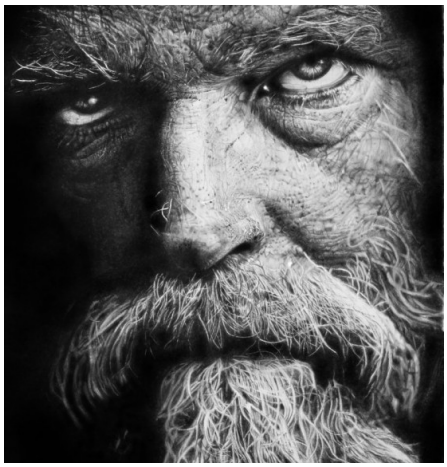

Figure 1: Drawing of a human face by Franco Clun. Details with relatively bigger depth (with their neighborhoods) are painted darker.

Other approaches can be considered as a simplification of the global illumination approach. They are usually designed to simulate some particular visual effects. This paper focuses on the screen-space ambient occlusion (SSAO) method that simulates ambient occlusion taking into account only visible parts of the virtual scene.

All techniques described above work well with conventional 2D displays. However, it is far from trivial to make them work on the state-of-the-art 3D visualization systems, like augmented or virtual reality (AR and VR), stereoscopic displays that work with glasses, multi-

view displays, or light field displays (LFDs). Light field displays – such as the HoloVizio system [Balazs14, Balogh07] – require much more computational power for rendering compared to the visualization on 2D displays. Because of this, graphics algorithms need to be as light-weight and as efficient as possible. Based on its attributes, the SSAO can be considered to be a fair solution for this type of systems.

This paper introduces the adaptation of SSAO algorithm of CryEngine 2 (see [Mitt07]) to the HoloVizio light field display, as well as some improvements for it.

The paper is structured as follows. Section 2 provides a brief overview of the state-of-the-art work performed in the topic of surface shading. This is followed by the overview of the HoloVizio light field visualization system in Section 3. The operation of the conventional SSAO technique is detailed in Section 4, followed by our contribution of a 3D SSAO extension in Section 5 and possible quality improvements in Section 6. Section 7 describes the performance of our algorithm. The paper is concluded in Section 8, also pointing out potential continuations of the investigated topic.

2 RELATED WORK

It is intuitively understandable that human observers can perceive the variations of depth by the level of darkness of the appropriate parts of the image. This fact was empirically proven [Lan00], together with some clarifications on the accuracy of subjective depth estimation under certain conditions. This evidence gives us the scientific justification of using the ambient occlusion family of algorithms to depict the variation of depth by the variation of light intensity.

Currently, the most precise method to depict the variation of light intensity in a given scene is to make a physically-based simulation. This approach is directly implemented in the global illumination (GI) frameworks, and it is widely applied in the creation of modern movies and cartoons [Chris10, Tab04]. However, using GI techniques is not always an optimal solution for real-time rendering – such as virtual and augmented reality, video games, or the real-time preview during the editing of 3D graphics – as these techniques come with higher requirements for computational power.

Another approach to render the photo-realistic images is to use the *ray tracing* family of methods. This includes the ray tracing method in its classic form [Wald09, Whit79], *path tracing* (also known as the Monte Carlo ray tracing [Veach97]), bidirectional path tracing [Laf93], etc. However, these mentioned approaches usually require additional computational power and necessitate the construction of additional acceleration structures (e.g., BVH, BIH, kd-trees, octrees, etc.). Also, despite of numerous algorithmic and hardware improvements, and the fact that they are already being widely applied offline, they are still only entering the market of existing real-time solutions at the time of this paper.

One way to avoid the unnecessary complexity of ray tracing algorithms in real-time frameworks is to trace the whole beam of light instead of a single ray. This method is called *cone tracing*, and can be applied in real-time to compute the illumination term only [Cras11].

Instead of doing a complete simulation of light behavior, one can try to imitate only a few necessary visual effects, including the AO effect. The mathematical background for these effects can be derived from the *rendering equation* [Kaj86], altogether with the other effects produced by the global illumination approach. Rather straightforward method to produce the AO effect in a given point of the virtual scene is to trace a limited number of rays in different directions from this point, and to check whether or not they intersect another surface of the scene (see Figure 2). In this case, the AO term is usually defined as the ratio of the number of rays that do not result in intersection to the total number of the emitted rays [Bun05, Dim08, Laine10, Shan07].

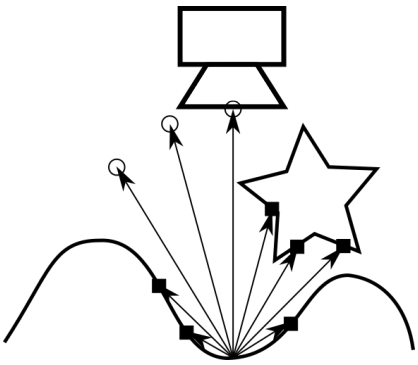

Figure 2: Conventional AO approach. Arrows represent rays emitted from the point of interest. Black squares designate the points that increase the AO value, white circles – that do not.

Several improvements and modifications have been made within the family of AO algorithms. Dimitrov et al. [Dim08] introduced the *horizon-split ambient occlusion*. Laine and Karras [Laine10] developed a ray-tracing-based method that takes into account the neighboring set of geometrical primitives (i.e., triangles). The evaluation of light intensity with the help of an additionally constructed simplified representation of the scene geometry in form of disks is introduced by Bunnell [Bun05], and by Shanmugam *et al.* [Shan07] – in the form of balls.

The main difference between the AO and SSAO algorithms is that the SSAO algorithm computes the AO value for visible surface only (see Figure 3), while conventional AO algorithm uses the whole scene for

this purpose. In conventional graphics pipeline, visible scene geometry can easily be constructed using additional depth texture that contains the information about the depth of each pixel.

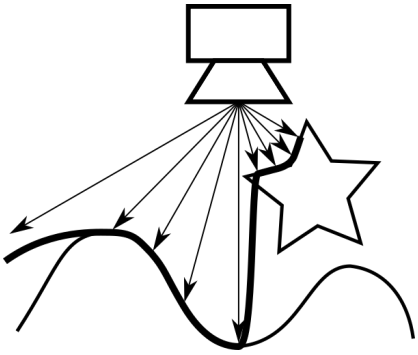

Figure 3: Approximation of the scene geometry by its visible parts (thick black line) used in the SSAO-family of algorithms.

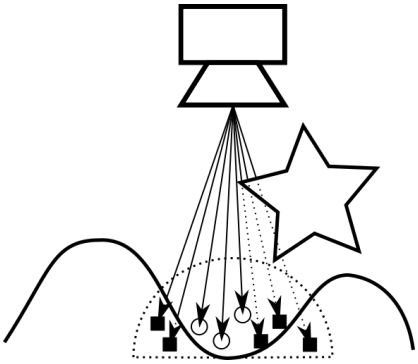

Figure 4: Screen-space ambient occlusion. Black squares designate the invisible samples, white circles are for the visible samples.

The collection of SSAO methods is implemented in CryEngine 2 [Mitt07], which is well-known for the Far Cry video game. In this approach, the random points (*samples*) are generated within a certain volume around the point of interest (*hit point*) on the visible surface. The AO value is defined through the ratio of invisible samples to the overall number of generated samples. Hoang and Low [Hoang10] suggested the *multi-resolution SSAO* that can be combined with other SSAO methods, the main idea of which is to measure the AO term for different radii and merge obtained information.

Loos and Sloan [Loos10] introduced the *volumetric obscurance* method (see Figure 5) that calculates the cumulative length of the unoccluded rays (depicted by thick gray lines on Figure 5) coming from the camera position instead of their number (as in the SSAO approach of Mittring [Mitt07]). The work of McGuire *et al.* [McGuire11] describes the application of a similar method in the industrial Alchemy framework.

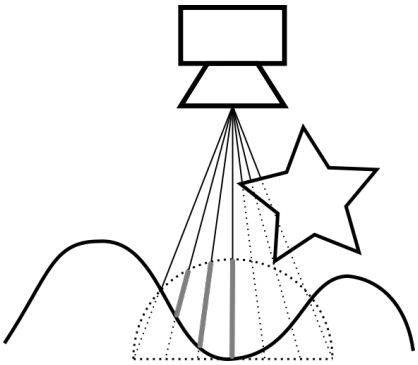

Figure 5: Volumetric obscurance.

3 OVERVIEW OF THE HOLOVIZIO SYSTEM

Probably, one of the most commonly accepted ways to parameterize light field is the 4D *two-plane parameterization* (see Figure 6). Each ray of light in light field is defined by the points of it's intersection with two parallel planes (*screen plane* and *observer plane*), and each point of intersection is parameterized by two coordinates.

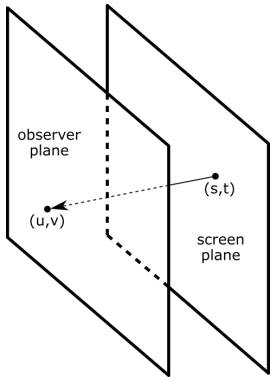

Figure 6: Two-plane 4D parameterization.

Two-plane parameterization is commonly used for full parallax light field displays. But for horizontal-only parallax displays (like HoloVizio system) the observer plane can be substituted by *observer line* [Balazs14] (see Figure 7). In this case, 4D two-plane parameterization can be replaced with 3D *plane-line parameterization* (two coordinates for the screen plane, one for the observer line). Analogously to two-plane parameterization, the observer line can be imagined as the possible positions of the viewer with respect to the actual screen.

Introduced plane-line parameterization is actually used in the most simple architecture of the HoloVizio system. In this form, HoloVizio system consists of actual physical screen (represented by the screen plane) with special diffuse properties, and a row of optical modules placed behind the screen. Each ray of light emitted by an optical module traverses through the screen without obstacles in horizontal direction, but diffused uniformly in all vertical directions to be able to hit the ob-

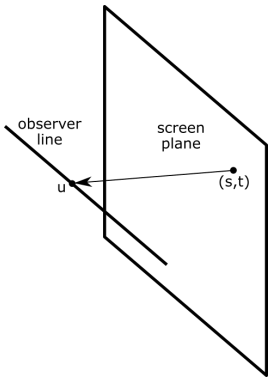

Figure 7: Horizontal parallax 3D parameterization.

server line (see Figure 8). More advanced architectures of the HoloVizio system may include convex display or different placing of optical modules, but the main concept remains the same. Real-life HoloVizio system may contain different number of optical modules (e.g., it is 80 for HoloVizio C80 cinema system).

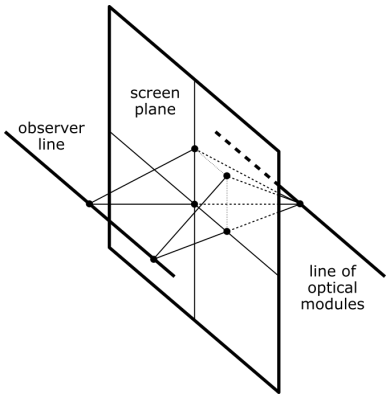

Figure 8: Distortion of particular rays of light coming from a single optical module.

For the practical reasons, we differentiate the following Euclidean 3D spaces: the *world space*, the *physical space*, and the *image space*. In the world space, the coordinates of the virtual scene are expressed. In the physical space, X axis is parallel to the observer line, Y axis goes vertically up, and Z axis is perpendicular to the screen plane. Physical space is introduced to simplify the calculation of the distortion of light that comes through the screen. Image space is defined for each optical module separately. In this space, the actual input 2D texture of the optical module is computed. X and Y axes in the image space are X and Y axes of the texture, and Z axis is the same as Z axis in the physical space.

The transformation from the world space to the physical space is affine and invertible. It can be set up by defining a perpendicular parallelepiped in each space (also known as the *region of interest*, ROI), and calculating the transformation from one to another. While the transformation from the physical space to the image space is not affine and not invertible. The relations between all three mentioned spaces is depicted on Figure 9.

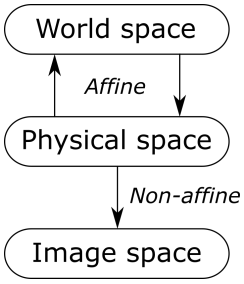

Figure 9: Relations between the world space, the physical space, and the image space.

Although the contribution of this paper is specifically made for HoloVizio light field displays, the introduced algorithm can be extended to other types of light field systems. The main reason to apply the introduced solution to a concrete system would be the necessity to use additional hierarchy of 3D spaces, similar to one depicted on Figure 9.

4 THE CONVENTIONAL SSAO ALGORITHM

In 2D case, the physical space does not exist, and the image space is substituted with the *screen space*. X and Y coordinates in the screen space are the same as X and Y coordinates of the pixel in the final image, and Z coordinate (*depth*) corresponds to the distance from the viewer to the corresponding point of the virtual scene. The exact way of computation of Z coordinate depends on whether orthographic or perspective projection is used. Note that there is an invertible homogeneous transformation between the world space and the screen space. This transformation allows to easily determine the precise position of each visible point on the scene, if both pixel coordinates and depth are known.

Our algorithm is based on the approach published by Mittring [Mitt07] (see Algorithm 1). This algorithm can be implemented as an OpenGL compute shader that is executed per each pixel of the final image. It takes the depth texture as the input, as well as two 4-by-4 matrices that correspond to the world-to-screen and the screen-to-world transformations. First, it calculates the position of a point of the visible surface that corresponds to the current pixel in the screen and the world space coordinates. Then it spawns samples (random 3D points) within a predefined radius `R` of calculated point in the world space. Furthermore, there is a check for each sample whether it is visible (i.e., positioned in front of the visible surface) or not. If the sample is not visible, the algorithm adds a value to the AO variable. In our implementation, the value of AO variable varies from zero to one.

Algorithm 1 Conventional SSAO algorithm. Variables `pntScrn` and `smpScrn` represent coordinates in the screen space, variables `pntWrld` and `smpWrld` – in the world space.

```
1:  pntScrn.xy ← texture coordinate of given pixel
2:  pntScrn.z ← depthTexture(pntScrn.xy)
3:  pntWrld ← screen-to-world(pntScrn)
4:  S ← set of random samples around pntWrld
5:  for each sample from S do
6:      smpWrld ← world coordinates of sample
7:      smpScrn ← world-to-screen(smpWrld)
8:      surDepth ← depthTexture(smpScrn.xy)
9:      if surDepth < smpScrn.z then
10:         increase AO variable
```

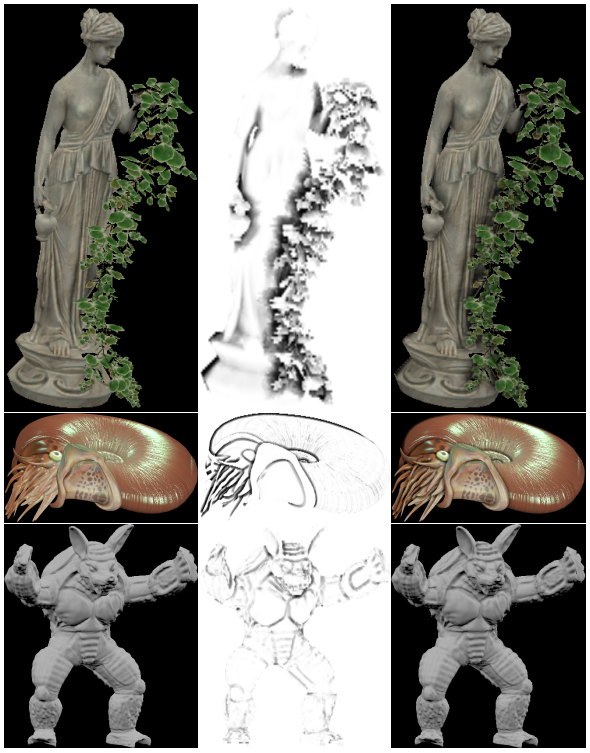

Figure 10: Rendered models of statue (top), ammonite (center) and armadillo (bottom) for one optical module. The left column represents the images without the SSAO effect. Images in the right column are after the SSAO effect was applied. The grayscale images with $(1-\texttt{ao})$ values are in the center.

The easiest way to apply the desired AO effect is to multiply each channel of the resulting RGB image by the $(1-\texttt{ao})$ term, where `ao` is the appropriate value in the computed AO texture. Figure 10 shows an example of this effect.

5 THE THREE-DIMENSIONAL SSAO ALGORITHM

The concept of the image space in HoloVizio system is very close to the concept of the screen space in conventional 2D graphics pipeline. X and Y coordinates of a point in each of these spaces are the pixel coordinate of the output texture of the shading algorithm, while Z coordinate represents the notion of depth in it's particular way. The main problem of application of the algorithm introduced by Mittring [Mitt07] is that the image-to-world transformation (analogous to screen-to-world transformation from algorithm 1) cannot be easily defined for the HoloVizio system.

The solution of the mentioned problem is to compute the world-space positions of the visible surface before the application of SSAO algorithm, and to store them in a separate texture (*positions texture*). In this case, the application of the image-to-world transformation for a point `pntImg` on the visible surface in the image space, can be substituted with getting the value of positions texture at `pntImg`.*xy* coordinates (symbol *.xy* designates the first two coordinates of a 3D vector). The final modification of the algorithm is presented in Algorithm 2.

Algorithm 2 SSAO algorithm for HoloVizio. Variables `pntImg` and `smpImg` represent coordinates in the image space, variables `pntWrld` and `smpWrld` – in the world space.

```
1:  pntImg.xy ← texture coordinate of given pixel
2:  pntWrld ← positionsTexture(pScn.xy)
3:  S ← set of random samples around pntWrld
4:  for each sample from S do
5:      smpWrld ← world coordinates of sample
6:      smpImg ← world-to-image(smpWrld)
7:      surDepth ← depthTexture(smpImg.xy)
8:      if surDepth < smpImg.z then
9:          increase AO variable
```

Algorithm 2 looks very similar to the Algorithm 1. The most noticeable difference is that rows 2 and 3 of Algorithm 1 are substituted with row 2 in Algorithm 2. However, there also are two other important distinctions. First, the world-to-image transformation in Algorithm 2 also includes the physical-to-image transformation described in Section 3, which is not invertible and cannot be expressed as a simple combination of homogeneous 4-by-4 matrices. Second, depth texture used in Algorithm 2 contains values obtained with the help of the mentioned world-to-image transformation. Alternatively, taking a value from the depth texture (row 7 in Algorithm 2) can be substituted with two operations: taking the world-space coordinates from positions texture, and applying world-to-image transformation to these coordinates.

6 IMPROVING THE QUALITY

It is possible to improve the quality of the introduced algorithm by the following methods.

Generation of samples

In most of SSAO implementations, samples are generated as a set of points inside a unit sphere (or a ball). Perhaps the most straightforward way to do this is to generate the points with uniform distribution within a bigger volume (cube), and to use the rejection method to leave only the necessary ones. However, we consider this method not to be flexible enough. The main reason for this is that on a small number of samples (around 8) the generated set can be not very well represented in all parts of the sphere.

In our implementation, we use the cylindrical parameterization to generate points inside the sphere uniformly. Algorithm 3 shows the usage of this parameterization. To use this approach to implement the conventional random uniform distribution on the spherical surface, one have to substitute the values ξ_1 and ξ_2 in this algorithm with the uniformly distributed random values from -1 to 1.

Algorithm 3 Generation of points on the spherical surface in cylindrical coordinates.

1: $\xi_1, \xi_2 \leftarrow$ values from -1 to 1;
2: $z \leftarrow \xi_1$;
3: $\varphi \leftarrow \pi\xi_2$;
4: $r \leftarrow \sqrt{1-z^2}$;
5: $x \leftarrow r\cos\varphi$;
6: $y \leftarrow r\sin\varphi$;

Algorithm 4 introduces the way to generate the coordinates of the point with uniform distribution in the inner volume of the ball with a given radius R, based on pregenerated points on the unit sphere. Thus, to get the uniformly distributed set of samples within a ball of radius R, one has to supply Algorithm 3 with ξ_1 and ξ_2 uniformly distributed on $[-1,1]$, and Algorithm 4 with $r_{min} \leftarrow 0$ and $r_{max} \leftarrow R$. In our implementation, we found that the choice $r_{max} \leftarrow R$ and $r_{min} \leftarrow 0.8 \cdot R$ is suitable enough.

Algorithm 4 Generation of random radius uniformly inside the given boundaries.

1: $x, y, z \leftarrow$ coordinates of a point on the unit sphere (output from Algorithm 3);
2: $r_{min}, r_{max} \leftarrow$ min and max boundaries of the radius;
3: $t \leftarrow$ uniformly distributed value from r_{min}^3 to r_{max}^3;
4: $r \leftarrow \sqrt[3]{t}$;
5: $x \leftarrow r \cdot x; \quad y \leftarrow r \cdot y; \quad z \leftarrow r \cdot z;$

The most flexible part in the introduced way to generate random samples using cylindrical coordinates lies in different methods to choose ξ_1 and ξ_2 values from Algorithm 3. One can easily notice that the parametric space of ξ_1 and ξ_2 values is nothing more than the $[-1,1] \times [-1,1]$ square in ordinary case. Thus, one can try to generate the random points in this parametric space in more sophisticated ways. We implemented the jittered sampling (subdividing space into grid cells and generating equal number of uniformly distributed points inside each cell), the Latin square approach (subdividing space into grid cells and generating only one random sample in each row and column), and the Poisson disk sampling [Brid07]. Although, we have not yet performed any subjective test that would reflect the significant difference in the perceived quality of the resulting images for each approach.

Additional blur pass

The algorithmic complexity of the introduced SSAO approach depends linearly on the number of samples being used. The common method to speed up the algorithm is to use a smaller number of samples, but compensate it with proper placement (e.g., by the methods described in the previous subsection). We found that it is reasonable to use approximately 8 samples. However, if we use a small number of samples, some visual artifacts in the final image may appear. To avoid these artifacts, we can do an additional Gaussian blur pass with a small kernel radius after the SSAO texture is rendered (see Figure 11).

Figure 11: Ambient texture before (left) and after (right) the blur pass was applied. Note that the artifacts in form of single white pixels on the left image that disappear after the blur pass.

Multi-resolution SSAO

There are usually many details of different size and shape in the virtual scene. This means that when the SSAO algorithm is applied, the radius of the sphere which incorporates the random samples should depend on the surrounding geometry. However, using additional information about the needed level of detail would add some complexity to the algorithm, making it slower in the end. To solve this problem, we implemented the *multi-resolution* ambient occlusion algorithm (see Figures 12, 13) introduced by Hoang and Low [Hoang10]. The main idea of this solution is to calculate the AO term for a couple of different radii (5 in our framework) and to merge the obtained information (we take the maximal AO contribution).

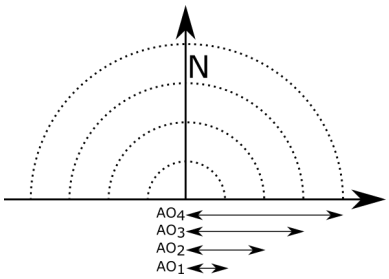

Figure 12: Multiresolution ambient occlusion. Labels $AO_1, ..., AO_4$ depict the areas in which the AO term is computed separately.

Figure 13: Ambient texture with single (left) and multiple AO radii (right).

7 PERFORMANCE

All mentioned techniques were implemented in a single framework for the HoloVizio system. For this purpose, we used the deferred rendering scheme. In total, there are six render passes for each frame:

1. Z prepass – preliminary checking of visibility;
2. G-buffer – generation of diffuse color texture, position texture, etc.;
3. SSAO texture generation;
4. horizontal Gaussian blur for SSAO texture;
5. vertical Gaussian blur for SSAO texture;
6. post processing – application of lights and SSAO effect.

We measured performance of the HoloVizio framework on a machine with Intel i7-5820K 3.30GHz CPU and GeForce GTX 960 video card. The tested HoloVizio system was virtual, with only one optical module of 1024-by-768 resolution. The rendered models are shown on Figure 10, and include:

1. *statue* (5011 faces, 3 textures);
2. *ammonite* (29520 faces, 4 textures);
3. *armadillo* (212574 faces, 0 textures).

For each model, we calculated the average time to render a single frame as the arithmetic average of rendering time of 1000 frames (see Table 1) with and without the application of the SSAO algorithm. Results from Table 1 show that the time to apply the SSAO effect itself can be considered as constant (for one HoloVizio system), and in the tested virtual system it equals to approximately 6.45 milliseconds.

Model	No SSAO	SSAO	Difference
Statue	0.79	7.25	6.46
Ammonite	1.62	8.05	6.43
Armadillo	2.48	8.96	6.48

Table 1: Average time to render a single frame in milliseconds.

An example of the model of statue displayed on a real HoloVizio system with the SSAO effect is shown on Figure 14. The photo displayed in the figure was captured by a regular digital camera.

Figure 14: The model of statue rendered on the HoloVizio 640RC system[1]. Left image is rendered without SSAO effect, right image – with. Note that the images above are rendered with the help of multiple optical modules, not a single one. These images are not the same with the images for the optical modules from Figure 10.

8 CONCLUSIONS

In this paper, we proposed the modification of the method used for generating the SSAO effect, in order to make it suitable for the state-of-the-art HoloVizio light field system, and described several ways to improve its quality and performance. This algorithm can also be applied to different light field systems or systems that aim to generate immersive 3D environments

[1] Mentioned model of HoloVizio system is out of the market now, and the closest analogue to it is the HoloVizio 722RC system by the time of this paper.

(e.g., virtual reality, augmented reality, stereoscopic displays, etc.). Further research on this topic shall include modifications of other techniques (e.g., global illumination) of conventional 2D frameworks to make them compatible with light field systems and 3D visualization in general.

9 ACKNOWLEDGEMENTS

This project has received funding from the European Union's Horizon 2020 research and innovation programme under the Marie Sklodowska-Curie grant agreement No 676401. The work in this paper was funded from the European Union's Horizon 2020 research and innovation program under the Marie Sklodowska-Curie grant agreement No 643072, Network QoE-Net.

10 REFERENCES

[Balazs14] Balázs, Á., Barsi, A., and Kovács, P. T., and Balogh, T. Towards mixed reality applications on light-field displays, in 3DTV-Conference: The True Vision-Capture, Transmission and Display of 3D Video (3DTV-CON), 2014, pp. 1–4, IEEE, 2014.

[Balogh07] Balogh, T., Kovács, P.T., and Megyesi, Z. Holovizio 3d display system, in Proceedings of the First International Conference on Immersive Telecommunications, in ImmersCom '07 (ICST), Brussels, Belgium), pp. 1–5, ICST (Institute for Computer Sciences, Social-Informatics and Telecommunications Engineering), 2007.

[Brid07] Bridson, R. Fast poisson disk sampling in arbitrary dimensions., in SIGGRAPH sketches, p. 22, 2007.

[Bun05] Bunnell, M. Dynamic ambient occlusion and indirect lighting. Gpu gems, Vol. 2, No. 2, pp. 223–233, 2005.

[Chris10] Christensen, P.H. Point-based global illumination for movie production, in ACM SIGGRAPH 2010 Course Notes, 2010.

[Cras11] Crassin, C., Neyret, F., Sainz, M., Green, S., and Eisemann, E. Interactive indirect illumination using voxel cone tracing, in Computer Graphics Forum, Vol. 30, pp. 1921–1930, Wiley Online Library, 2011.

[Dim08] Dimitrov, R., Bavoil, L., and Sainz, M. Horizon-split ambient occlusion, in Proceedings of the 2008 Symposium on Interactive 3D Graphics and Games, I3D '08, (New York, NY, USA), p. 1, ACM, 2008.

[Hoang10] Hoang, T.D., and Low, K.L. Multi-resolution screen-space ambient occlusion, in Proceedings of the 17th ACM Symposium on Virtual Reality Software and Technology, VRST '10, New York, USA, pp. 101–102, ACM, 2010.

[Kaj86] Kajiya, J.T. The rendering equation, in ACM Siggraph Computer Graphics, Vol. 20, pp. 143–150, ACM, 1986.

[Laf93] Lafortune, E.P., and Willems, Y.D. Bi-directional path tracing, in Proceedings of the 3rd International Conference on Computational Graphics and Visualization Techniques (Compugraphics), Vol. 10, (Alvor, Portugal), pp. 145–153, 1993.

[Laine10] Laine, S., and Karras, T. Two methods for fast ray-cast ambient occlusion, in Proceedings of the 21st Eurographics Conference on Rendering, EGSR'10, Aire-la-Ville, Switzerland, pp. 1325–1333, Eurographics Association, 2010.

[Lan00] Langer, M.S., and Bülthoff, H.H. Depth discrimination from shading under diffuse lighting. Perception, Vol. 29, No. 6, pp. 649–660, 2000.

[Loos10] Loos, B.J., and Sloan, P.P. Volumetric obscurance, in Proceedings of the 2010 ACM SIGGRAPH symposium on Interactive 3D Graphics and Games, pp. 151–156, ACM, 2010.

[McGuire11] McGuire, M., Osman, B., Bukowski, M., and Hennessy, P. The alchemy screen-space ambient obscurance algorithm, in Proceedings of the ACM SIGGRAPH Symposium on High Performance Graphics, pp. 25–32, ACM, 2011.

[Mitt07] Mittring, M. Finding next gen: Cryengine 2, in ACM SIGGRAPH 2007 Courses, SIGGRAPH '07, New York, USA, pp. 97–121, ACM, 2007.

[Shan07] Shanmugam, P., and Arikan, O. Hardware accelerated ambient occlusion techniques on gpus, in Proceedings of the 2007 Symposium on Interactive 3D Graphics and Games, I3D '07, New York, USA, pp. 73–80, ACM, 2007.

[Tab04] Tabellion, E., and Lamorlette, A. An approximate global illumination system for computer generated films. ACM Trans. Graph., Vol. 23, pp. 469–476, Aug. 2004.

[Veach97] Veach, E., and Guibas, L.J. Metropolis light transport, in Proceedings of the 24th annual conference on Computer graphics and interactive techniques, pp. 65–76, ACM Press/Addison-Wesley Publishing Co., 1997.

[Wald09] Wald, I., Mark, W. R., Günther, J., Boulos, S., Ize, T., Hunt, W., Parker, S.G., and Shirley, P. State of the art in ray tracing animated scenes, in Computer Graphics Forum, Vol. 28, pp. 1691–1722, Wiley Online Library, 2009.

[Whit79] Whitted, T. An improved illumination model for shaded display, in ACM SIGGRAPH Computer Graphics, Vol. 13, p. 14, ACM, 1979.