IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Date of publication xxxx 00, 0000, date of current version xxxx 00, 0000.

Digital Object Identifier xxx/ACCESS.2020.DOI

DeepAO: Efficient Screen Space Ambient Occlusion Generation via Deep Network

DONGJIU ZHANG[1], CHUHUA XIAN[1], GUOLIANG LUO[2], YUNHUI XIONG[3], and CHU HAN[4]

[1]Guangdong Provincial Key Lab of Computational Intelligence and Cyberspace Information, School of Computer Science and Engineering, South China University of Technology, Guangzhou 510006, China (e-mail: chhxian@scut.edu.cn)
[2]East China Jiaotong University, Nanchang, China. (e-mail: luoguoliang@ecjtu.edu.cn)
[3]School of Mathematics, South China University of Technology, Guangzhou 510006, China (email: yhxiong@scut.edu.cn)
[4]Department of Radiology, Guangdong Provincial People's Hospital, Guangdong Academy of Medical Sciences, Guangzhou 510080, China (email: zq1992@gmail.com)

Corresponding author: Chuhua Xian (e-mail: chhxian@scut.edu.cn) and Guoliang Luo (e-mail: luoguoliang@ecjtu.edu.cn).

This work is jointly supported by the Natural Science Foundation of Guangdong Province under Grant 2019A1515011793 and 2017A030313347, the National Natural Science Foundation of China under Grant 61962021 and 51978271, the China Postdoctoral Science Foundation under Grant 2019M662261, and the Key Research and Development Program of Jiangxi Province under Grant 20192BBE50079.

ABSTRACT Ambient occlusion (abbr. AO) plays an important role in realistic rendering applications because AO produces more realistic ambient lighting, which is achieved by calculating the brightness of certain screen parts based on objects' geometry. However, the baseline computation of AO algorithm is time-consuming, which limits its application for real-time rendering. Currently, most AO algorithms are based on screen space to reduce the computational consumption, which leads to unrealistic results due to the usage of artificial features. To overcome these challenges, in this paper, we first create a well-crafted dataset with the pair of deferred shading buffer data and ground-truth AO shaded images. Then, we design an efficient deep neural network for the screen space AO image generation, based on which we further design a Compute Shader Library to compute the shaded AO images. Our extensive experimental results show that our method achieves competent performance than existing screen space ambient or volumetric ambient based AO methods both in visual quality and efficiency.

INDEX TERMS Ambient Occlusion, Rendering, Shading, Deep Neural Network

I. INTRODUCTION

AMBIENT Occlusion (abbr. AO) plays a vital role in the lighting of a scene, because it determines the percentages of light being 'blocked' by the environment. The original AO is a complex ray tracing process that requires high computation cost, which normally works as a preprocess in applications [1], [2]. To reduce the computation cost, there are two approximation methods: the screen space ambient occlusion (abbr. SSAO) [3]–[7], and the volumetric ambient occlusion (abbr. VAO) [8]–[10]. SSAO-based methods utilize the screen space information, including depth and normal, stored in G buffer to compute the occlusions that are independent from the complexity of scene. These methods are extensively used in real-time applications because of the simple implementation and high efficiency. Different with SSAO, VAO-based methods transform the sampling domain to a smaller tangent sphere and formulate the AO problem as a volumetric integration. However, these existing methods often produce incorrect results due to the artificial features.

In recent years, many methods based on neural network have been proposed in the domain [11], [12]. The major features of these methods are based on a dataset, and then train neural network to learn a mapping from the input data surrounding the pixel to the AO of that pixel. However, due to the deficiency in network structures, these methods are difficult to fully learn full information in samples, i.e., the generalization of their trained models are not satisfactory. With the development of the graphic hardware, although the ray-traced AO methods may run in real-time with the advanced graphic cards, they are also limited due to the high cost for regular application scenarios [13].

In this paper, we propose an efficient SSAO method,

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/ACCESS.2020.2984771, IEEE Access

FIGURE 1. Example scenes rendered with our **DeepAO** model enabled (bottom row) and disabled (top row). All results are rendered within Unity game engine.

named **DeepAO**, to generate accurate AO map via deep neural network. We first create a dataset that contains camera space depth, normal and ground truth AO that can be computed via a ray-trace-based render. Then, we design a light U-Net [14]-like fully convolution neural network to learning the mapping between the input G-buffer information and the ground truth AO shaded images on this dataset. By comparing with the existing machine learning based methods, our proposed method achieves a better accuracy and more efficient in run-time. Fig. 1 shows four example scenes generated by using DeepAO. Based on our method, we have also developed a Compute Shading Library, which is simple, fast and efficient implementation of neural network in game engine. This library can be easily applied to other rendering tasks.

Instead of defeating the Ray-traced AO method on high performance computers, our motivation in this work is to generate high quality AO results efficiently on mid-and-low performance devices. In summarize, our main contribution are as follows:

- We have proposed an improved structure to generate comparable results with that of using off-line Ray-traced method efficiently.The source codes and our Shading Library have been released at [15].
- We have presented a more complex dataset, which can better support the generalization of the training based methods.

II. RELATED WORK

AO is a perennial topic in computer graphics, and a range of approaches have been proposed on this task. Early methods use ray-tracing algorithm to generate still images [1], [2], [16]. Subsequently these methods are used as a preprocessing step in real-time interactive applications though limited to static scene. To support the dynamic scenes, Kontkanen et al. [17] applied the ambient occlusion fields to handle animated scenes [18]. Bunnnell [19] presents a GPU-based method for deformation surface. In [20], Chistensen extends this method for rendering color bleeding. All these mentioned methods are based on ray tracing or various surface discretization.

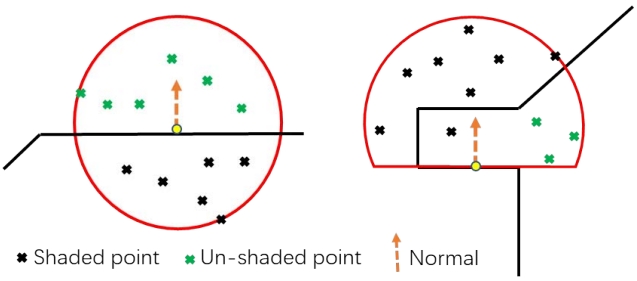


FIGURE 2. Two sampling methods of the SSAO-based models: Sampling in a view spheric space (left) and sampling in a hemispheric space (right).

Dating from 2007, the feasible methods to compute ambient occlusion in real-time along with other affect in virtual world named screen-space ambient occlusion (abbr. SSAO) methods are developed [3]–[7]. These methods sample from depth buffer in a view sphere space and count the number of occluded points to estimate the occlusion factor (Fig. 2 (a)). To address the self-occlude flaw, the SSAO+ [21] method changes spherical space to hemispheric space, as illustrated in Fig. 2 (b). This greatly improves the performance and make SSAO-based methods to work within dynamic environ-

ments in conjunction with animation and physics. Moreover, SSAO-based methods do not require pre-calculated occlusion data but relieves both the artists and the memory budget. Base on SSAO, some extended methods have been proposed. In [22], Bavoi et al. introduce Horizon Based Ambient Occlusion (abbr. HBAO). This method computes the occlusion by estimating the openness of the horizon about the sample point. Then the rays are marched along the depth buffer, and the horizon estimation are calculated by the difference in depth. However, this method only takes into account of the visible points in the current tracing perspective, and does not integrate the global influence. As shown in Fig. 3, the points that are invisible in current view also make a contribution for the ambient occlusion. Moreover, due to the sampling mechanism of HBAO, it is difficult to deal with points that are relatively far away in the sampling space. These drawbacks will lead to incorrect shadows or the absence of shadows.

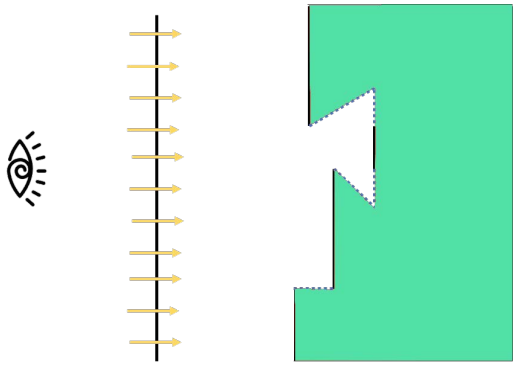


FIGURE 3. An example of the invisible geometric information from the perspective of Z-buffer. The points in dot are also make contribution for AO. In this case, the HBAO method does not take these points into consideration.

Different with SSAO, another technique called Volumetric Ambient Occlusion(abbr. VAO) is proposed [10], [23]. This methods measures the portion of the tangent sphere of the surface belonging to the set of occluded points and replaces the ray trace by containment tests. The integrated new formula has low variation that allows to be estimated accurately with a few samples. However, due to the differences of the pure AO and VAO, the VAO methods can only produce overly smooth results which lack of details of the scene.

Besides of SSAO and VAO, Penmatsa et al. propose the VXAO method in [24]. Instead of relying on screen space data, this method gathers information from a world space in voxel representation of the scene, which covers a large area around the viewer. In this representation, objects that are relatively far from the surface still have contributions for the occlusion. However, in order to reduce the computational cost, this method requires to use the lower resolution of the voxels that leads to a low resolution shading result and cannot guarantee the rendering quality, which limits its applications. Moreover, the computation cost remains more expensive than the HBAO method even though it samples the space in a low resolution.

With the success applications of deep learning, many researchers have studied deep learning based methods to treat the rendering problems. In [25], Yang et al. use a Deep Convolution Networks to perform a quick reconstruction for Monte Carlo Rendering. To address the AO problem, Holden et al. [11] first introduce a neural network, named NNAO, to learn the features from dataset and predict an optimal approximation of the ambient occlusion affect. The network of NNAO is a multi-layer-perception with only 4 layers, for which the network has poor feature learning ability and cannot generate high quality AO results. Afterwards, Nalbach et al. [12] introduce a convolution neural network named Deep Shading to learn a mapping from position in camera space and normal to AO shaded image. However, the Deep Shading method have poor performance on generalization due to the low variety of objects of the training dataset.

Recently, real-time AO methods have been developed with special graphic hardware support, such as NVIDIA RTX 2080ti [13]. Although the RTX graphic card can achieve 1080P in real-time, the fps may vary in 30-60, depending on the complexity of the scene. However, with a device does not have an NVIDIA RTX Graphic card, such as the mobile device, the method cannot generate high quality AO in real-time. In contrast, SSAO-based method can efficiently run on the devices with OpenGL 3.1 (or later version) support, which allows broad usage in common applications.

III. PROPOSED METHOD

A. DATASET GENERATION

There exists two public datasets of AO: the NNAO [11] and the Deep Shading [12] datasets. However, the NNAO dataset only contains about 600 annotated records that lacks the variety of objects, which limits the generalization of the trained model. For the Deep Shading dataset, the geometry details of the scenes are relatively coarse, which cannot facilitate to guarantee the quality of the AO results.

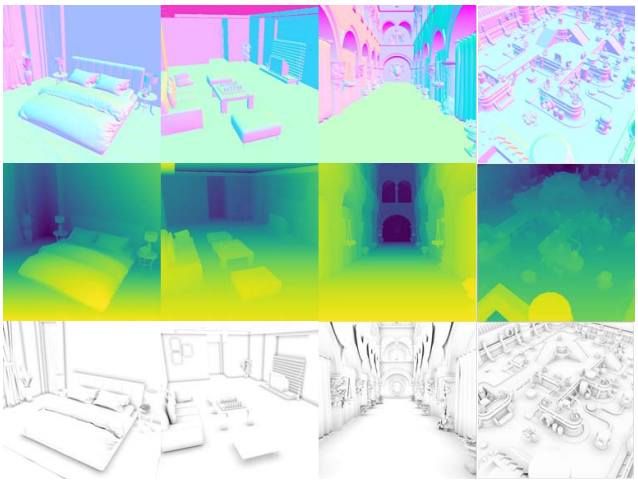


FIGURE 4. Exampled scenes in our dataset. The first row shows the normal maps, the second row are the corresponding depth maps. The corresponding ground truth AO shaded images are shown in the bottom.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/ACCESS.2020.2984771, IEEE Access

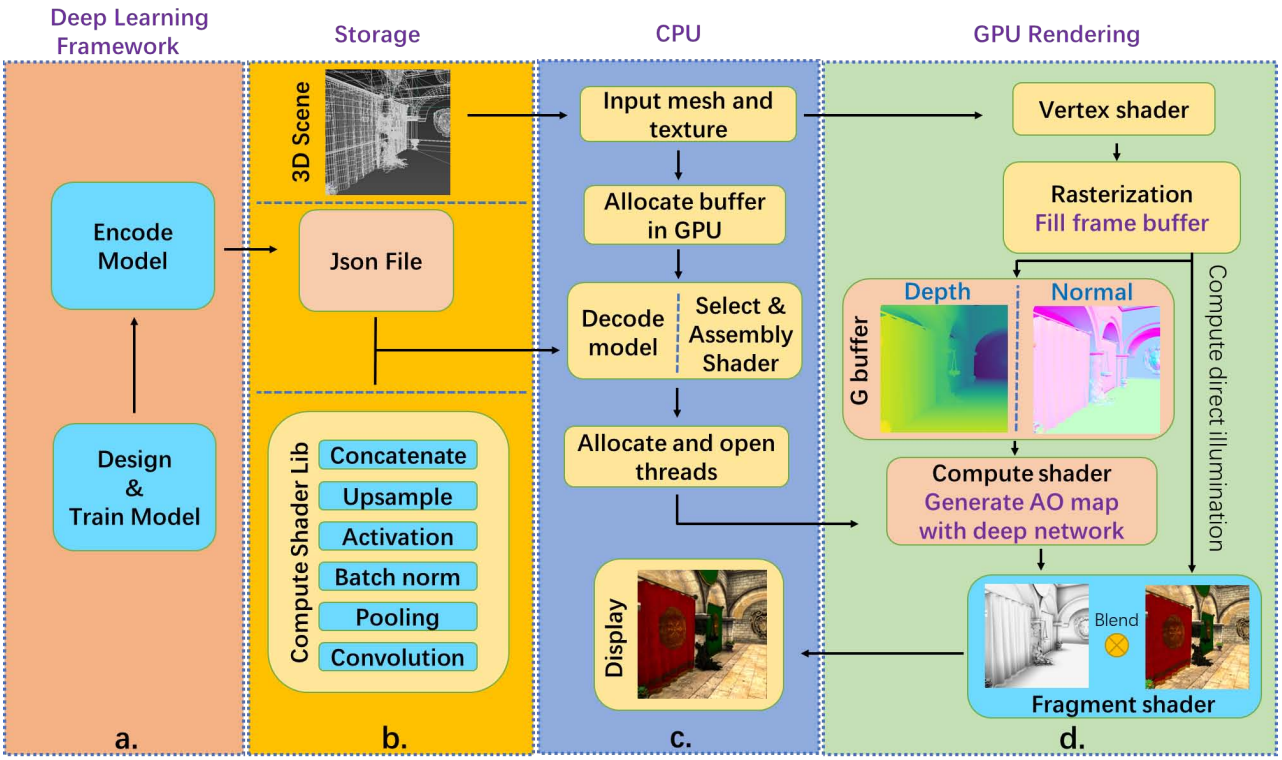


FIGURE 5. The workflow of our **DeepAO** model, which consists of four modules: the deep learning module, the storage module, the CPU computation module and the GPU rendering module.

To overcome the above limitations, we build our own dataset with both deferred shading buffers and ground truth AO shaded images. To create the dataset, 17 scenes with more than 200 classifies of objects are utilized to generate the pair annotated data. Each of the scene contains a wide variety of objects with high resolution of geometry details, such as car, ship, aircraft, building, status, etc. Among these scenes, 15 are used to generate the training data and 2 are used for generating the validate data.

In total, our dataset contains $42,000$ pairs of deferred shading buffers with their ground truth AO shaded images in a resolution of 512×512 px. The ground truth AO shaded images are generated by the professional rendering engine (V-Ray [26]) using the ray tracing algorithm offline. To generate the $42,000$ AO shaded images, we create $6-8$ virtual cameras for each scene and set the field of view to 50 degrees for all cameras. Then, we rotate each camera with $90, 180$, and 270 degrees perpendicular to the angle of view, and bind them to the pre-designed paths and make them shooting along the paths. It took more than five days for us to render all scene data with the ray-traced V-Ray engine on two PCs with high performance GPU devices offline. The size of the generated dataset is more than 500 G. We will release our collected scenes and the codes including the shading library for generating the dataset publicly upon this work is accepted. We believe researchers will benefit to explore further studies on AO.

It worth to point out that the deferred shading buffers which include the depth map and the normal map are generated based on camera space. Fig. 4 shows four annotated pairs in our dataset. There are two reasons: 1) the normal and depth map of camera space can be retrieved directly from the G-buffer when delayed rendering is enabled, 2) we cannot find the additional benefits of using normal vector and position in world coordinates.

In this work, we use $33,800$ annotated pairs to train our proposed deep network model on our dataset, and use $7,200$ pairs to validate the trained model. Besides, we also test the generalization of our trained model using the common household scenes and famous public scenes such as Sponza Palace in our experiment (see Fig. 7), which are not included in training dataset.

B. WORKFLOW OF DEEP AO

Holden et al. [11] and Nalbach et al. [12] both designed a pixel shader that integrated all the implementations. In their frameworks, a user is required to modify the implementation of the shader if there are updates of the AO algorithms. In order to improve the workflow to more flexible and independent of the neural network, we propose to divide the workflow into several independent modules.

As illustrated in Fig. 5, our workflow mainly consists of four independent modules: the deep learning module, the storage module, the CPU computing module, and the GPU

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/ACCESS.2020.2984771, IEEE Access

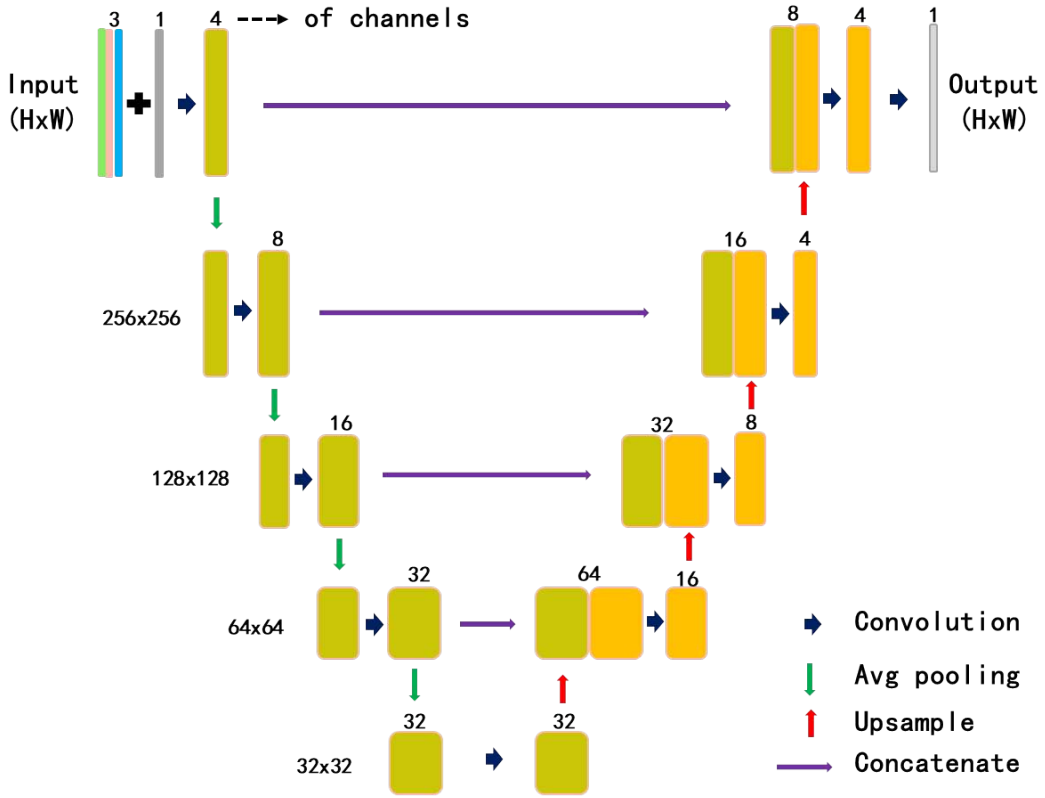


FIGURE 6. The network structure of the proposed AO generation model. The input is the depth map and camera space normal map at the resolution of $W \times H$. The resolution of the output is the the same as the input.

rendering module. With the powerful parallel computation capability of GPU device, we design six parameterized compute shaders with the operations of convolution, batch norm, activation, pooling, up-sample, and concatenate in our task. We package these shaders in a library, shown in Fig. 5(b). This library includes encode-decode models and reflection operations. With this Compute Shader Library, the pre-saved model can be automatically decoded and re-generated by conducting a different deep learning framework without updating any code, even if the network structure is modified. Therefore, users only need to focus on the design of deep learning frameworks in this workflow. To reduce the runtime of our algorithm, we also provide further encapsulated shader that combines all the operations into few shaders. This structure reduces the number of drawcalls and greatly improves the rendering efficiency. Similar to the other screen-space ambient occlusion methods, the input of our method includes the normal vector and the depth which can be easily obtained in G-buffer when delay rendering is enabled.

C. NETWORK

Our method utilizes the network structure of U-Net which is commonly applied for the medical image segmentation [14], [27]. As illustrated in Fig. 6, the proposed networks includes the down-sampling branch on the left and the up-sampling branch on the right. The down-sampling branch is a network which consists of four 3×3 convolution kernel with stride 1 and Leaky-ReLU active layers, and four 2×2 average-pooling structure with stride 2. Each step of the up-sampling branch consists of a bi-linear upper sampling layer, a convolution (reducing the number of channels by half) layer, batch normalization, and leaky-ReLU activation layer. The last layer uses a 1×1 convolution kernel to map four channels to a single channel. It worth to mention that although the resolution of the shaded images in our dataset is 512×512, our proposed network is based on the full convolutional network, which can output the same size as the input at any arbitrary resolution.

Loss Function. In the training step of the network, we take structure similarity (abbr. SSIM) index as the loss function with a window size of 11. The output of the last layer is $x_i \in [0, 1]$, and $y_i \in [0, 1]$ is the real sample tag. Then, we define the loss function as follows:

$$SSIM(x, y) = \frac{(2u_x u_y + c_1)(2\sigma_{xy} + c_2)}{(u_x^2 + u_y^2 + c_1)(\sigma_x^2 + \sigma_y^2 + c_2)} \quad (1)$$

where u_x and u_y are the mean of the patch; σ_x, σ_y and σ_{xy} are the variances. $c_1 > 0$ and $c_2 > 0$ are constants which are used to avoid errors of dividing by zero. We set $c_1 = 1e-4, c_2 = 9e-4$ in our implementation.

Training Parameters. In the training step, the Adaptive Moment Estimation [28] is used in our network, and the

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/ACCESS.2020.2984771, IEEE Access

learning rate is 0.0001. In our implementation, the batch size is set to 16, and the leaky-ReLU negative slope is set to 0.01.

IV. EXPERIMENTS

The implementation platform of our experiment is PyCharm with PyTorch port. All the experiments in this paper are conducted on a PC with an Intel(R) Core(TM) i7-9700 CPU at 3.6G and a NVIDIA GeForce GTX 1080ti graphic card. The memory is 16G, and the OS is 64-bit Windows 10 with professional version. All the test are measured with Unity Profiler.

Quantitative Evaluation: To evaluate the performance of our method, we tested our trained model on three datasets: the NNAO dataset [11], the deep shading dataset [12], and our new dataset. The statistical results of our verification are given in Table 1. We also compare our method with three state-of-the-art SSAO-based methods: the HBAO [6], the NNAO [11], and the Deep Shading method [12]. The average values for the structure similarity (abbr. SSIM) index and the root-mean-square error (abbr. RMSE) are reported. We also test the average run-time on each dataset. The best performance is marked in bold in the table. As can be seen, our method outperforms all other three compared methods on these three datasets.

TABLE 1. Comparisons of our model with three SSAO-based methods. The average RMSE, average SSIM and average run-time of the methods are computed on the datasets: NNAO [11], Deep Shading [12] and our new dataset.

DATASETS	NNAO		DEEP SHADING		OURs		Run-time(ms)
	SSIM	RMSE	SSIM	RMSE	SSIM	RMSE	
HBAO	0.867	0.072	0.737	0.063	0.821	0.058	6.19
NNAO	0.45	0.156	0.354	0.133	0.36	0.117	3.41
Deep Shading	0.873	0.151	0.711	0.082	0.875	0.077	12.5
DeepAO	**0.908**	**0.059**	**0.745**	**0.061**	**0.899**	**0.046**	**3.13**

We have also conducted a set of visual comparisons, see Fig. 7. The original NNAO method adopted the bilateral filtering to post-process the output, because its original results contain lots of noise. Thus, in our experiment, only the NNAO output is processed with the bilateral filtering operations for a fair comparison. As can be seen, our method shows significantly better results than other three methods: although the HBAO method yields good results in smaller structure, it generates poor results in larger structure, especially when there are partial occlusions. The results generated by the NNAO method have poor shadows in many regions. It worth to point out that the number of the sampling points in original NNAO in [11] is 8, and the AO results of the NNAO method highly depends on the sampling points. That is, with more sampling points input that along with more computation time, NNAO shows better AO results. For a fair comparison, we set the input sample number up to 64 in all comparisons. As shown in Table 1, our method achieves better performance both in visual quality and run-time. This is because the network of NNAO is a simple multi-layer perception with only 4 neural layers which strongly limits its learning ability.

Furthermore, comparing to the deep shading method [12] , our method also shows better performance it both in visual effectiveness and run-time. We analyze the reasons as follows:

- The deep shading method requires to input the camera coordinates (P_x, P_y, P_z) to compute the transformation from depth to the camera coordinates. On the contrary, our method directly reads the depth value from the Z-buffer, for which the computation becomes more efficient.
- The convolution kernels of down sampling step in the network of deep shading method are $8, 16, 32, 64$, and 128, respectively. Using the conventional convolution will be up to 40 ms for one input. To accelerate the computation, the deep shading method adopts the group convolution operations. In our experiments, we found that the conventional convolution can generate better results. To balance the effectiveness and the efficiency, we use the convolution kernels with $4, 8, 16, 32$ in the down sampling step, respectively.
- The ground truth AO shaded images in deep shading dataset contain many noise, and the loss function of SSIM is sensitive to these noise. For this reason, the learning ability of deep shading method is strongly limit. Moreover, the variety of the scenes for generating the deep shading dataset is simple and the scenes do not contain high resolution mesh models, which further limits the generalization of deep shading method for other complex scenes. In obvious contrast to the deep shading dataset, the collected scenes of our new datasets contain a large amount of different objects in high resolutions. This variety greatly improves the generalization of our method. It worth to point out that we re-trained the deep shading network on our new dataset and re-trained our proposed network on deep shading dataset for fair comparisons.

Comparison with VAO-based method: VAO is another technique for generating AO [10], [23]. In our experiments, we compare our method with the latest VAO-based method: VAO++ [10]. Fig. 8 shows two examples of comparison results. The VAO++ results are generated by the addin in Unity [29]. As can be seen, the visual quality of our results is much better than the VAO++ method. Additionally, the numeric measurements of SSIM and RMSE also outperforms the VAO++ method.

V. CONCLUSION

In this paper, we propose an efficient screen space ambient occlusion generation method via deep network. We first construct our dataset with pairs of deferred shading buffers and ground truth shaded AO images. Then, we design a U-Net-like network to train on the dataset to generate the ambient occlusions. We evaluate our method by comparing with the existing traditional and deep-learning based methods

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/ACCESS.2020.2984771, IEEE Access

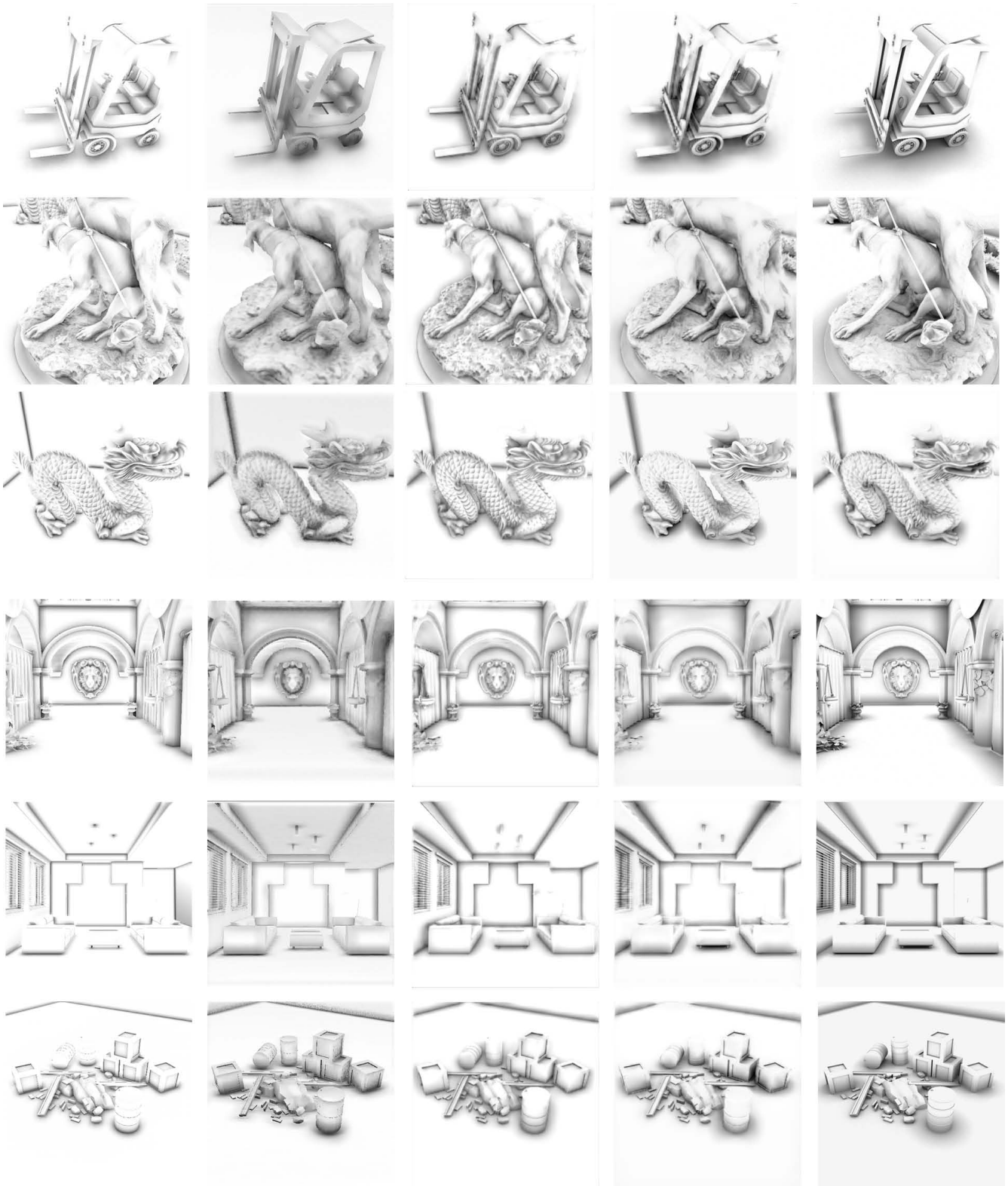


FIGURE 7. Comparisons with the existing SSAO-based methods. From left to right: HBAO [6], NNAO [11], Deep shading [12], our model, and the ground truth. Note that the example in the fourth row is from the Sponza Palace scene, which is not included in either the training or validation. that only the NNAO output is conducted by the bilateral filtering operation to alleviate the noise.

on both two public datasets and our datasets. Experimental results show that our method achieve better performance both in visual quality and run-time than these methods on the three datasets. Furthermore, we also improve the workflow

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/ACCESS.2020.2984771, IEEE Access

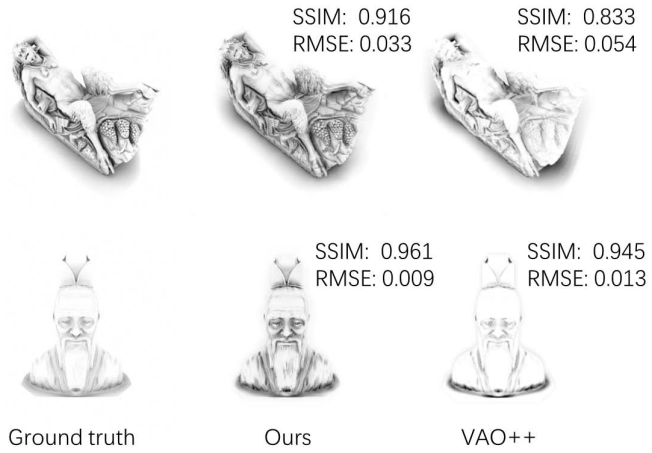
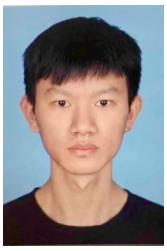


FIGURE 8. Comparisons with the VAO++ method [10]. From left to right: ground truth, our model, and the VAO++ method. Corresponding SSIM and RMSE are shown on the top-right corner of each shaded AO result.

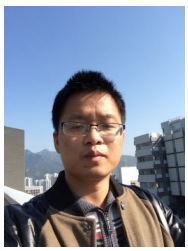
and design a Compute Shader Library which contains common convolution network operation, encode-decode model, and reflection. This allows to automatically generate neural network shader that can produce fast accurate results. Our Compute Shader Library will be easy to for users to add into the existing rendering pipeline as a post-process step according to the configuration file stored in the frozen model.

REFERENCES

[1] S. Zhukov, A. Iones, and G. Kronin, "An ambient light illumination model," in Rendering Techniques' 98. Springer, 1998, pp. 45–55.
[2] H. Landis, "Production-ready global illumination," Siggraph course notes, vol. 16, no. 2002, p. 11, 2002.
[3] M. Mittring, "Finding next gen: Cryengine 2," in ACM SIGGRAPH 2007 courses. ACM, 2007, pp. 97–121.
[4] H. Nguyen, Gpu gems 3. Addison-Wesley Professional, 2007.
[5] P. Shanmugam and O. Arikan, "Hardware accelerated ambient occlusion techniques on gpus," in Proceedings of the 2007 symposium on Interactive 3D graphics and games. ACM, 2007, pp. 73–80.
[6] L. Bavoil and M. Sainz, "Screen space ambient occlusion," NVIDIA developer information: http://developers. nvidia. com, vol. 6, 2008.
[7] P. Clarberg and T. Akenine-Möller, "Exploiting visibility correlation in direct illumination," in Computer Graphics Forum, vol. 27, no. 4. Wiley Online Library, 2008, pp. 1125–1136.
[8] L. Szirmay-Kalos, T. Umenhoffer, B. Tóth, L. Szécsi, and M. Sbert, "Volumetric ambient occlusion for real-time rendering and games," IEEE Computer Graphics and Applications, vol. 30, no. 1, pp. 70–79, 2009.
[9] M. Ruiz, L. Szirmay-Kalos, T. Umenhoffer, I. Boada, M. Feixas, and M. Sbert, "Volumetric ambient occlusion for volumetric models," The Visual Computer, vol. 26, no. 6-8, pp. 687–695, 2010.
[10] J. Bokšanskỳ, A. Pospíšil, and J. Bittner, "Vao++: practical volumetric ambient occlusion for games," in Proceedings of the Eurographics Symposium on Rendering: Experimental Ideas & Implementations. Eurographics Association, 2017, pp. 31–39.
[11] D. Holden, J. Saito, and T. Komura, "Neural network ambient occlusion." in SIGGRAPH Asia Technical Briefs, 2016, p. 9.
[12] O. Nalbach, E. Arabadzhiyska, D. Mehta, H.-P. Seidel, and T. Ritschel, "Deep shading: convolutional neural networks for screen space shading," in Computer graphics forum, vol. 36, no. 4. Wiley Online Library, 2017, pp. 65–78.
[13] E. Liu, I. Llamas, P. Kelly et al., "Cinematic rendering in ue4 with real-time ray tracing and denoising," in Ray Tracing Gems. Springer, 2019, pp. 289–319.
[14] X. Wu and L. Zhao, "Study on iris segmentation algorithm based on dense u-net," IEEE Access, vol. 7, pp. 123 959–123 968, 2019.
[15] https://github.com/chuhuaxian/DeepAO.
[16] A. Iones, A. Krupkin, M. Sbert, and S. Zhukov, "Fast, realistic lighting for video games," IEEE computer graphics and applications, vol. 23, no. 3, pp. 54–64, 2003.
[17] J. Kontkanen and S. Laine, "Ambient occlusion fields," in Proceedings of the 2005 symposium on Interactive 3D graphics and games. ACM, 2005, pp. 41–48.
[18] J. Kontkanen and T. Aila, "Ambient occlusion for animated characters." in Rendering techniques. Citeseer, 2006, pp. 343–348.
[19] M. Bunnell, "Dynamic ambient occlusion and indirect lighting," Gpu gems, vol. 2, no. 2, pp. 223–233, 2005.
[20] P. Christensen, "Point-based approximate color bleeding," Pixar Technical Notes, vol. 2, no. 5, p. 6, 2008.
[21] https://www.moddb.com/news/introducing-ssao-and-loot-system.
[22] L. Bavoil, M. Sainz, and R. Dimitrov, "Image-space horizon-based ambient occlusion," in ACM SIGGRAPH 2008 talks. ACM, 2008, p. 22.
[23] L. Szirmay-Kalos, T. Umenhoffer, B. Tóth, L. Szécsi, and M. Sbert, "Volumetric ambient occlusion for real-time rendering and games," IEEE Computer Graphics and Applications, vol. 30, no. 1, pp. 70–79, 2010.
[24] R. Penmatsa and C. Wyman, "Voxel-space ambient occlusion," IOWA UNIV IOWA CITY DEPT OF COMPUTER SCIENCE, Tech. Rep., 2012.
[25] X. Yang, D. Wang, W. Hu, L. Zhao, X. Piao, D. Zhou, Q. Zhang, B. Yin, Q. Cai, and X. Wei, "Fast reconstruction for monte carlo rendering using deep convolutional networks," IEEE Access, vol. 7, pp. 21 177–21 187, 2018.
[26] https://vray.us/.
[27] O. Ronneberger, P. Fischer, and T. Brox, "U-net: Convolutional networks for biomedical image segmentation," in International Conference on Medical image computing and computer-assisted intervention. Springer, 2015, pp. 234–241.
[28] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," arXiv preprint arXiv:1412.6980, 2014.
[29] https://projectwilberforce.github.io/vaopaper/.



DONGJIU ZHANG is currently a master candidate in the School of Computer Science and Engineering at South China University of Technology. His current research interests include computer graphics, image processing, and computer vision.



CHUHUA XIAN is currently an associate professor in the School of Computer Science and Engineering at South China University of Technology. He has a PhD in Computer Science from the State Key Lab of CAD&CG at Zhejiang University in 2012. He was a postdoctoral researcher in CUHK from Nov. 2013-May 2014 and from Sep. 2015-Apr. 2016. His current research interests include intelligent computer graphics, image processing, geometry modeling and processing, and 3D computer vision.

GUOLIANG LUO earned his PhD in Computer Science at University of Strasbourg in 2015. His research interests include computer graphics, artificial intelligence. He is currently an associate professor at East China Jiaotong University. He was enrolled in the Ganjiang Outstanding Youth Talent Program in 2018.

YUNHUI XIONG is an associate professor in School of Mathematics at South China University of Technology. He received Ph.D. degree in computer science engineering from the South China University of Technology in 2010. His research interests include computer graphics, distributed source coding, geometry modeling and processing, 3D reconstruction and 3D printing.

CHU HAN is now a postdoctoral fellow at the Guangdong Provincial People's Hospital, Guangdong Academy of Medical Sciences, under the supervision of Prof. Zaiyi Liu and Prof. Changhong Liang.He received his Ph.D. degree from the Chinese University of Hong Kong, under the supervision of Prof. Tien-Tsin Wong. He received the M.Sc. degree in computer science from South China University of Technology, and the B.Sc. degree from South China Agricultural University. His current research interests include medical image analysis, computer graphics, image processing, computer vision and deep learning.