Ambient Occlusion Baking via a Feed-Forward Neural Network

Ugo Erra, Nicola Capece, and Roberto Agatiello

Università della Basilicata, Dipartimento di Matematica, Informatica ed Economia, Italy

Abstract

We present a feed-forward neural network approach for ambient occlusion baking in real-time rendering. The idea is based on implementing a multi-layer perceptron that allows a general encoding via regression and an efficient decoding via a simple GPU fragment shader. The non-linear nature of multi-layer perceptrons makes them suitable and effective for capturing non-linearities described by ambient occlusion values. A multi-layer perceptron is also random-accessible, has a compact size, and can be evaluated efficiently on the GPU. We illustrate our approach of screen-space ambient occlusion based on neural network including its quality, size, and run-time speed.

Categories and Subject Descriptors (according to ACM CCS): I.3.3 [Picture/Image Generation]: Display Algorithms—I.3.7 [Three-Dimensional Graphics and Realism]: Color, shading, shadowing, and texture—

1. Introduction

Shadowing of ambient light is called ambient occlusion. It has been shown in [LB00] that ambient occlusion offers a better perception of the 3D shape of displayed objects, and its effectiveness is evident in its popularity in videogame engines [Mit07]. The mathematical definition of ambient occlusion is related to the concept of the solid angle. In fact, the occlusion $A_{\mathbf{p}}$ at a point $\mathbf{p}$ on a surface with normal $\mathbf{n}$ can be computed by integrating the visibility function over the hemisphere Ω with respect to the projected solid angle:

$$A_{\mathbf{p}} = \frac{1}{\pi}\int_{\Omega} V_{\mathbf{p},\omega}(\mathbf{n}\cdot\omega)\,d\omega \qquad (1)$$

where $V_{\mathbf{p},\omega}$ is the visibility function at $\mathbf{p}$ along a direction ω. A simple method to approximate this integral in practice, in off-line rendering, is based on ray-tracing. Rays are shot in a uniform pattern across the hemisphere over point $\mathbf{p}$, and an occlusion value can be calculated as the number of rays that hit the geometry divided by the total number of rays shot. Rays can be restricted to a certain length, avoiding the need to take into account distant geometry while calculating the occlusion value. This is fundamental in closed environments, which would otherwise result in total occlusion at every point and subsequently the complete removal of ambient light.

Videogames often precompute ambient occlusion and bake out the results into vertex or texture data, which is later loaded into OpenGL or DirectX shaders. To implement such a baking pipeline, an implementation of ray-tracing is used as off-line rendering. When the ambient occlusion values have been computed for a 3D shape, they must be written in a format that can be easily consumed during the rendering. The methods used in film and game rendering are point clouds, 2D textures, and vertex attributes. Point clouds are not efficient to access in hardware for real-time rendering. 2D textures retain the detail from the original ambient occlusion values, but require too many memory resources. Vertex attributes require occlusion values from the 3D surface shape to be mapped onto the vertices. The last solution yields significantly lower memory overhead and less expensive run-time reconstruction for real-time applications such as videogames [KBS11].

Our key idea is to represent ambient occlusion by using a multi-layer perceptron, allowing general encoding via regression and efficient decoding via a simple GPU fragment shader in real-time. Because of the non-linear nature of multi-layer perceptrons, they are suitable and effective for capturing non-linearities described by ambient occlusion values. In addition, multi-layer perceptrons are random-accessible, have a compact size, and can be evaluated efficiently on the GPU. Our approach offers benefits in terms of quality, size, and run-time speed particularly for low-poly models used in real-time application. To evaluate the quality of results, we compare screenshots from our implementation with images rendered off-line in the ray-tracing engine NVIDIA OptiX [PBD*10], which is also used to train our neural network. For comparison, we use the structural similarity (SSIM) index, which is a metric that measures similarity between images in a way that is consistent with human eye perception.

2. Related Works

Real-time global illumination is a hot topic in computer graphics research, and an impressive number of related works have been published. Here, we restrict the scope to techniques most closely related to our own: ambient occlusion suitable to real-time rendering that operates in image space (also called screen-space ambient

DOI: 10.2312/egsh.20171003

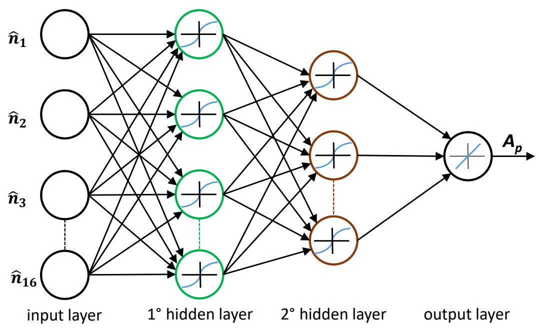

Figure 1: *The acyclic feed-forward neural network, which defines a mapping from 16 sample normals in object space to the output ambient occlusion. We evaluated three neural network with different hidden layers: NN-*16×8*, NN-*32×16*, and NN-*64×32*.*

occlusion). The technique called "screen-space ambient occlusion" appears for the first time in [Mit07]. In its CryEngine, CryTek implements this technique, which works by sampling the surroundings of a pixel and, on the basis of the z-buffer, performs depth comparisons. Sample positions are distributed in a sphere around the pixel, and some randomness is introduced by reflecting position vectors on a random plane passing through the sphere origin. The occlusion factor depends only on the depth difference between sampled points and the current point. This, combined with the simple distribution of samples (around a sphere and not a hemisphere) causes some over-darkening: even flat, non-occluded areas result in some samples being considered as occluders. Even in only this category of algorithms, a variety of approaches exist. Some sources attempt to correlate, compare, and evaluate such techniques, and the interested reader may like to consult [AB13] and [Gra13], which are two recent theses that agree that the Alchemy algorithm [MOBH11] was state of the art at that time. More recently, the state of the art in research is Mara et al. [MMNL16], which has been demonstrated to be an order of magnitude faster and more correct.

Neural networks have been used in real-time global illumination in [RWG*13]. In that work, the authors model a radiance regression function for fast rendering of global illumination in scenes with dynamic local light sources. The regression function is implemented by using a multi-layer acyclic feed-forward neural network, which provides a close functional approximation of the indirect illumination and can be efficiently evaluated at run time. A similar work based on a feed-forward neural network is [HSK16]. In this work, the authors train a neural network to learn a mapping from the depth and normals surrounding the pixel to the ambient occlusion. In the present work, we use only normals surrounding the pixel as the input of the neural network. The idea is to train the neural network to learn the shape of the surface to guess the occlusion value for a given surface point.

3. Our Approach

Here we present details of our feed-forward neural network, including representation, training/encoding, and rendering/decoding.

Representation. A neural network is a weighted and directed graph whose nodes are organized into layers. The weights of the edges constitute the components of the weight vector w. The network we use is an acyclic feed-forward network with two hidden layers, as shown in Figure 1. Each node is connected to all nodes of the preceding layer by directed edges, such that a node in the ith layer receives inputs from all nodes in the $(i-1)$th layer. The graph takes inputs through the nodes in the first layer, called the input layer, and produces outputs through the nodes in the final layer, called the output layer. The nodes in the input layer correspond to 16 normals in object space. The output layer consists of one node that is the value of the ambient occlusion. The layers between the input and output layers are called the hidden layers. In our work, we use a $16 \times X \times Y \times 1$ neural network, where $X \times Y$ are the two hidden layers. In particular, we evaluated three hidden layers: 16×8, 32×16, and 64×32.

Each neuron in a particular layer is connected with all neurons of the previous layer. The connection between a neuron k in the $(i-1)$th layer and a neuron j in the ith layer is characterized by the weight coefficients w_{kj}^{i-1}. Let n_j^i be the output of the node j in the ith layer and w_{j0}^i be its bias weight. Each node output of an hidden layer i is calculated from the outputs of all nodes in the $(i-1)$th layer as follows:

$$n_j^i = \sigma(z_j^i), \quad z_j^i = w_{j0}^i + \sum_{k>0} w_{kj}^{i-1} n_k^{i-1}$$

The summation is carried out over all neurons k transferring the signal to the ith layer. The function $\sigma(z_j^i)$ is the so-called transfer function and for all hidden layers is the hyperbolic tangent function $\sigma(z) = \tanh(z) = 2/(1+e^{-2z}) - 1$, while for the node in the output layer it is a linear function. The training process varies the bias weight w_{j0}^i and weight coefficients w_{kj}^{i-1} to minimize the sum of the squared differences between the computed and required output values.

Training. We use four models and select 128 points of views by using a rotating camera around the bounding sphere containing the model. From each point of view, we use OptiX [PBD*10] to render the global ambient occlusion at a resolution of 1920×1080. From each image, we randomly pick 512 pixels from which we obtain the ambient occlusion values. We then take 16 normal samples in the object space located around the ambient occlusion value, in a circle of radius 10. These values represent the input data for our neural network, while the ambient occlusion value of the central pixel is the corresponding output data. When completed, the final dataset has 65,536 data points. The dataset extracted with this approach produces the best results experimentally.

For the training, the dataset was split into three subsets: 50% as training set, 25% as validation set, and 25% as test set. Dataset splitting was done in a random way. The training was performed by using the scaled conjugate gradient back-propagation method [Møl93], and performance was evaluated by using the mean squared normalized error performance function. The number of epochs and max fail were set to 20,000, while the other parameters were set to default. The neural network was trained by using a NVIDIA Tesla K40c installed on a machine equipped with an Intel Core i7-3820 and 16 GB RAM. The average training time for the neural network of 16×8 hidden layers is about 4.50 minutes, for the network of 32×16 about 5.55 minutes, and for 64×32 about 7.53 minutes. Training was performed by using MATLAB's Neural Network Toolbox.

Rendering. After training the feed-forward neural network, the ambient occlusion of each point of the 3D mesh can be rendered in real-time by using the trained model in a fragment shader. We developed a shader by using OpenGL Shading Language, which is a straightforward translation of the neural network. Weights and biases are provided to the fragment shader by using the uniform buffer for allocation of GPU memory. In particular, the memory occupation for NN-16×8, NN-32×16, and NN-64×32 are 3.716 bytes, 8,452 bytes, and 20,996 bytes, respectively.

The sample normals are computed in the object space by using a G-buffer and then projecting them in the screen space by using a texture. In more detail, for a given viewpoint we first compute the visible surface points of each screen pixel. For each pixel, we take its normal and then assign it as input the same value over all inputs of the neural network. This way of querying the neural network model is different in terms of the way we trained it, but the approach produces excellent ambient occlusion without either noise or blurring artefacts, thus avoiding the cost of a Gaussian blur pass. This corresponds to taking 16 normal samples located around a pixel in a circle of radius zero.

4. Experimental Results

We implemented the rendering algorithm on a GPU via OpenGL API and GLSL shading language. All results and performance measures shown in this paper were conducted on a PC with 3.50 GHz i7-5930K CPU, 16GB memory, and NVIDIA GTX 1070 graphics card.

In Figure 2, we visually compare the results of our approach to the ground truth images rendered with OptiX, and Mara et al. (14 samples, 7.4 radius, 1 iteration, and 0.002 bias) [MMNL16]. These images are the final result of the fragment shader without any Gaussian blur pass. Compared with the ground truth images, where there are geometric creases and contact shadows between triangle meshes, our approach can produce a result that is less dark because of the radius used to sample normals. Our approach produces good results in general, and visually there are slight differences between the three neural networks as shown in Figure 4. In addition, because the training data are based only on the normals of the 3D model, depth resolution artefacts that occur in screen-space ambient occlusion techniques are not present regardless of the distance from the camera of the 3D model. To define the similarity between two images, we adopted the SSIM index [WBSS04] as a metric to measure similarity between images in a way that is consistent with human eye perception. In Table 2, we report the SSIM between images generated with the three neural networks and those generated with OptiX.

In Table 1, we show the results of a numerical comparison between our method and that of Mara et al. All measurements are taken at 1920×1080. The NN-16×8 offers better performances. For reference only, in the same table we also report the performance of rendering the same models by using vertex colours to store the baked ambient occlusion. We find that the NN-32×16 requires more computing resources, probably because of the loop performances, and NN-64×32 offers essentially the same performance. Nevertheless, we are confident that some optimizations can be implemented to increase performance.

5. Conclusions and Future Works

We present an approach for performing screen-space ambient occlusion using a feed-forward neural network. The training phase is performed by using only normals of the 3D model in object space and by using a rendering framework such as OptiX to create a high-quality ambient occlusion. We create a fragment shader that computes the values of the ambient occlusion in real-time efficiently and with accurate results. The proposed approach offers a new way to use precomputed ambient occlusion during rendering for geometry from low- to medium-grained high-frequency structures. The limit of this approach depends on the type of 3D model. Models with extreme variability and geometric complexity could cause network overfitting of the training data and fail to capture the distribution of the ambient occlusion over the model surface. Future works will focus on how to design dynamically the neural network based on geometric complexity. We will also study how to sample in a more efficient and effective way normals and the ambient occlusion values to avoid the selection of points of view when picking the pixels.

Acknowledgements

The authors thank NVIDIA's Academic Research Team for providing the Tesla K40c card under the Hardware Donation Program.

References

[AB13] AALUND F. P., BÃĘRENTZEN J. A.: *A Comparative Study of Screen-Space Ambient Occlusion Methods*. Bachelor thesis, Technical University of Denmark, Informatics and Mathematical Modelling, 2013. 2

[Gra13] GRAVÅS L. O.: *Image-space Ambient Obscurance in WebGL*. Tech. rep., Institutt for datateknikk og informasjonsvitenskap, 2013. 2

[HSK16] HOLDEN D., SAITO J., KOMURA T.: Neural network ambient occlusion. In *SIGGRAPH ASIA 2016 Technical Briefs* (New York, NY, USA, 2016), SA '16, ACM, pp. 9:1–9:4. 2

[KBS11] KAVAN L., BARGTEIL A. W., SLOAN P.-P.: Least squares vertex baking. In *Proc. of the Twenty-second Eurographics Conference on Rendering* (Aire-la-Ville, Switzerland, 2011), EGSR '11, Eurographics Association, pp. 1319–1326. 1

[LB00] LANGER M. S., BÜLTHOFF H. H.: Depth discrimination from shading under diffuse lighting. *Perception 29*, 6 (2000), 649–660. 1

[Mit07] MITTRING M.: Finding next gen: Cryengine 2. In *ACM SIGGRAPH 2007 Courses* (New York, NY, USA, 2007), SIGGRAPH '07, ACM, pp. 97–121. 1, 2

[MMNL16] MARA M., MCGUIRE M., NOWROUZEZAHRAI D., LUEBKE D.: Deep g-buffers for stable global illumination approximation. In *Proc. of High Performance Graphics* (Aire-la-Ville, Switzerland, 2016), HPG '16, Eurographics Association, pp. 87–98. 2, 3, 4

[MOBH11] MCGUIRE M., OSMAN B., BUKOWSKI M., HENNESSY P.: The alchemy screen-space ambient obscurance algorithm. In *Proc. of the ACM SIGGRAPH Symposium on High Performance Graphics* (New York, NY, USA, 2011), HPG '11, ACM, pp. 25–32. 2

[Møl93] MØLLER M. F.: A scaled conjugate gradient algorithm for fast supervised learning. *Neural networks 6*, 4 (1993), 525–533. 2

MODEL	NN-16 × 8	NN-32 × 16	NN-64 × 32	Mara et al.	Vertex colours
Bunny	1.96 ms	21.31 ms	29.29 ms	16.99 ms	0.19 ms
Buddha	2.74 ms	22.18 ms	29.97 ms	17.00 ms	0.67 ms
Dragon	2.17 ms	16.35 ms	22.76 ms	16.99 ms	0.55 ms
Lucy	2.25 ms	21.71 ms	29.54 ms	16.99 ms	0.32 ms

Table 1: *Comparison between the three neural networks, Mara et al. [MMNL16], and vertex attributes rendering.*

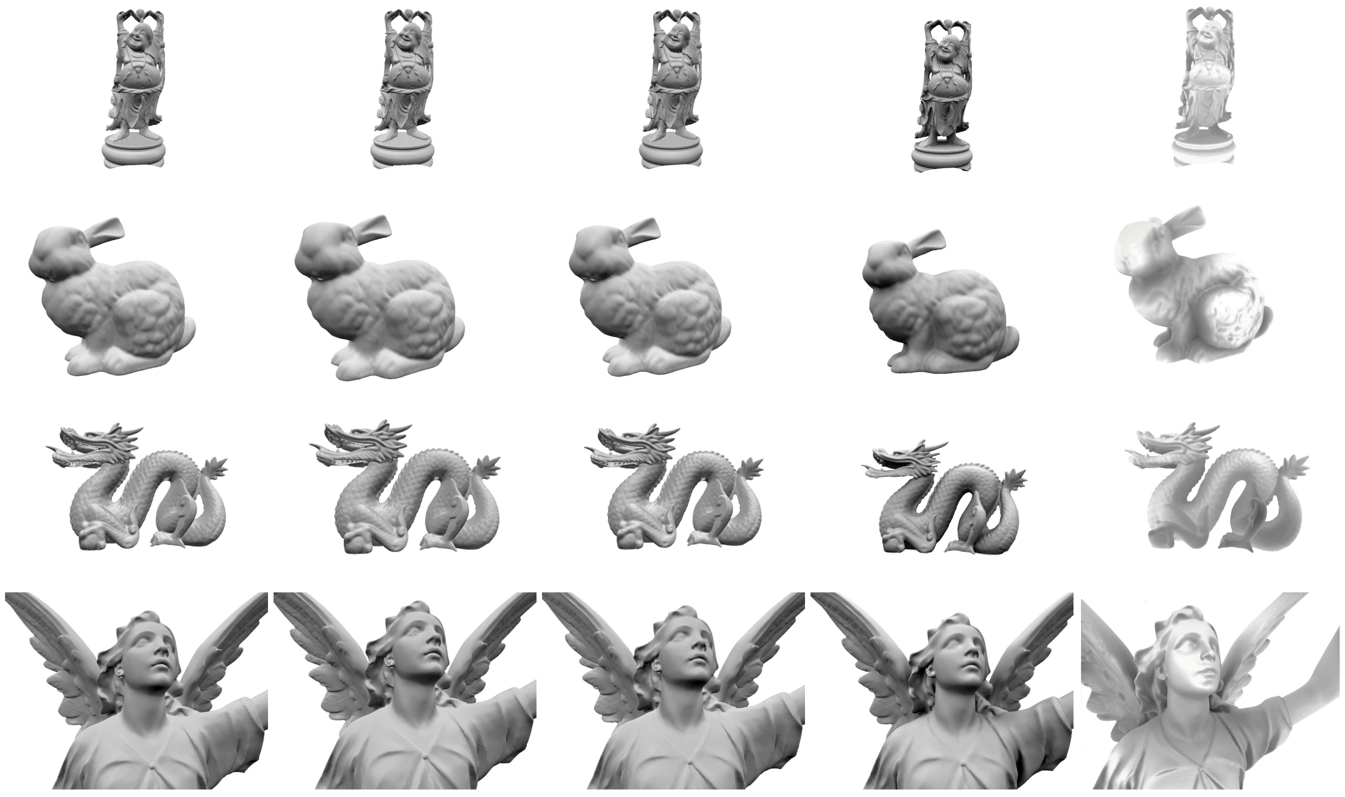

Figure 2: *From left to right: NN-16×8, NN-32×16, NN-64×32, ground truth with OptiX, and Mara et al. [MMNL16].*

MODEL	NN-16 × 8	NN-32 × 16	NN-64 × 32
Buddha	0.9646	0.9642	0.9655
Bunny	0.9814	0.9819	0.9793
Dragon	0.9495	0.9504	0.9489
Lucy	0.9743	0.9750	0.9754

Table 2: *Structural similarity (SSIM) index between images generated with the three neural networks and those generated in Figure 2 with OptiX.*

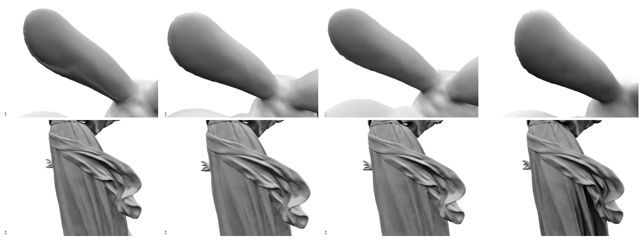

Figure 3: *Close-up of ambient occlusion results. From left to right: NN-16×8, NN-32×16, NN-64×32, and ground truth with OptiX.*

[PBD*10] PARKER S. G., BIGLER J., DIETRICH A., FRIEDRICH H., HOBEROCK J., LUEBKE D., MCALLISTER D., MCGUIRE M., MORLEY K., ROBISON A., STICH M.: Optix: A general purpose ray tracing engine. In *ACM SIGGRAPH 2010 Papers* (New York, NY, USA, 2010), SIGGRAPH '10, ACM, pp. 66:1–66:13. 1, 2

[RWG*13] REN P., WANG J., GONG M., LIN S., TONG X., GUO B.: Global illumination with radiance regression functions. *ACM Trans. Graph. 32*, 4 (July 2013), 130:1–130:12. 2

[WBSS04] WANG Z., BOVIK A. C., SHEIKH H. R., SIMONCELLI E. P.: Image quality assessment: from error visibility to structural similarity. *IEEE Transactions on Image Processing 13*, 4 (April 2004), 600–612. 3