Deep Real-time Volumetric Rendering Using Multi-feature Fusion

ACM Reference Format:
. 2023. Deep Real-time Volumetric Rendering Using Multi-feature Fusion. *ACM Trans. Graph.* 1, 1 (May 2023), 4 pages. https://doi.org/10.1145/3588432.3591493

A APPENDIX

A.1 More experiments

We present more comparison results between our model and the naïve RPNN, using the unbiased path tracer as the reference. All tests were carried out on an Nvidia RTX 2080 GPU with a 1024^2 resolution (exceptions explicitly noted). The resolution of volumetric data is 1024^3. “Cloud” refers to a cloud-shaped configuration, and “Model” to one created from a specified geometry.

Table 1: Bias and performance test results. Note that the neural network-based approaches are cost-consistent with change of light direction, whereas the reference (MC estimator) fluctuates in time. In the convergence test, we used 64SPP for both RPNN and MRPNN, and adaptive samples depending on the noise for the reference. Note that rendering takes much less time (≤ 0.5ms) than network inference.

Model	Light Dir.	Bias		Frame Cost (ms)		Convergence Boost	
		Ours	RPNN	Ours	RPNN	RPNN	Ref.
	Side	2.14e-2	2.49e-2				2687.5 ×
Cloud1	Front	1.97e-2	2.59e-2	5.0	584.0	116.8 ×	2203.1 ×
	Back	2.19e-2	3.22e-2				2937.5 ×
	Side	1.29e-2	1.46e-2				729.2 ×
Cloud2	Front	1.10e-2	0.87e-2	6.0	750.0	125.0 ×	677.1 ×
	Back	2.31e-2	1.86e-2				390.6 ×
	Side	1.96e-2	1.98e-2				2812.5 ×
Cloud3	Front	2.33e-2	1.63e-2	5.1	558.5	109.5 ×	2625.0 ×
	Back	3.69e-2	3.74e-2				2343.8 ×
	Side	1.65e-2	1.43e-2				1078.1 ×
Cloud4	Front	1.28e-2	1.07e-2	5.0	488.0	97.6 ×	984.4 ×
	Back	2.30e-2	4.95e-2				1050.0 ×
	Side	1.98e-2	3.44e-2				421.9 ×
Model1	Front	1.27e-2	0.94e-2	4.4	377.1	85.7 ×	375.0 ×
	Back	1.42e-2	1.85e-2				350.0 ×
	Side	1.54e-2	1.92e-2				386.4 ×
Model2	Front	1.01e-2	0.85e-2	5.5	650.1	118.2 ×	340.9 ×
	Back	1.82e-2	1.85e-2				306.8 ×

Author’s address:

0730-0301/2023/5-ART $15.00
https://doi.org/10.1145/3588432.3591493

Table 2: Check experiment of the biases with different shading parameters.

RMSE $\times 10^2$		Side		Front		Back	
Model	Parameter	Ours	RPNN	Ours	RPNN	Ours	RPNN
	$\varsigma = \{1.0, 1.0, 1.0\}$	2.14	2.49	1.97	2.59	2.19	3.22
Cloud1	$\varsigma = \{0.96, 0.98, 1.0\}$	1.59	n/a	1.36	n/a	2.12	n/a
	$\varsigma = \{0.8, 0.9, 1.0\}$	1.33	n/a	1.84	n/a	1.66	n/a
	$\varsigma = \{1.0, 1.0, 1.0\}$	1.29	1.46	1.10	0.87	2.31	1.86
Cloud2	$\varsigma = \{0.96, 0.98, 1.0\}$	1.14	n/a	0.98	n/a	2.26	n/a
	$\varsigma = \{0.8, 0.9, 1.0\}$	0.82	n/a	0.85	n/a	2.10	n/a
	$\varsigma = \{1.0, 1.0, 1.0\}$	1.98	3.44	1.27	0.94	1.42	1.85
Model1	$\varsigma = \{0.96, 0.98, 1.0\}$	1.64	n/a	1.23	n/a	1.43	n/a
	$\varsigma = \{0.8, 0.9, 1.0\}$	1.16	n/a	1.18	n/a	1.98	n/a
	$G = 0.857$	2.14	2.49	1.97	2.59	2.19	2.59
Cloud1	$G = 0.5$	2.51	n/a	2.02	n/a	1.64	n/a
	$G = 0.0$	2.40	n/a	1.67	n/a	2.12	n/a
	$G = 0.857$	1.29	1.46	1.10	0.87	2.31	0.87
Cloud2	$G = 0.5$	1.25	n/a	1.91	n/a	2.45	n/a
	$G = 0.0$	1.43	n/a	2.19	n/a	2.30	n/a
	$G = 0.857$	1.98	3.44	1.27	0.94	1.42	0.94
Model1	$G = 0.5$	2.03	n/a	0.79	n/a	0.95	n/a
	$G = 0.0$	2.27	n/a	0.88	n/a	1.25	n/a

A.2 Stencil pattern

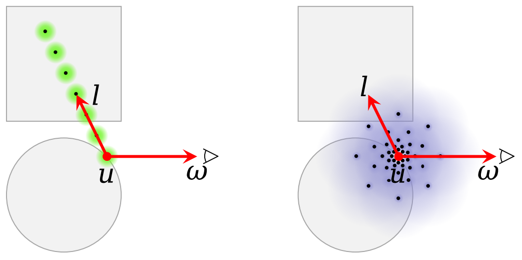
As shown in Fig. 1, we divide our stencil pattern into two halves, each focusing on high frequency (diffusion-) or low frequency (shadow-aware) information. Now we introduce the details of our frequency-sensitive stencil pattern.

The low-frequency part. The low-frequency part consists of spherical and intra-spherical distributed points. The *spherical* components refer to points spaced on a spherical surface, and the *intra-spherical* components denote points distributed uniformly inside a sphere, both radii denoted by r_i. The configuration of the low-frequency part is listed in Tab. 3, which totals 160 points. Specifically, Q_1 consists of 8 points where the first point lies in the center and the rest on the sphere.

Table 3: The configuration of the low-frequency layers.

Layer (i)	**Num. Points** (N_i)	**Mip-level** (m_i)	**Distribution**
1	8	0	central + spherical
2		1	spherical
3	16	2	
4		3	
5		4	
6	32	5	intra-spherical
7		6	
8		7	

The stencil points in each layer are obtained through an iterative relaxation algorithm. Since we wish the points to be as uniform as possible, the algorithm is based on maximizing the sum of the



(a) High-freq. part **(b) Low-freq. part**

Figure 1: Decomposing RPNN's stencil into high- and low-frequency part. We use spherical distributions instead of lattice grids in the low-frequency part.

Mutual Minimum Distance [Heinrich et al. 2008]:

$$R(Q_i) = \frac{1}{N_i} \sum_{\boldsymbol{p} \in Q_i} \min \{ \|\boldsymbol{p} - \boldsymbol{q}\|, \boldsymbol{q} \in Q_i \setminus \{\boldsymbol{p}\} \} . \tag{1}$$

Since the radii r_i at the current stage is unknown, we first consider auxiliary stencil layers $\hat{Q}_i$ where all the spheres are of a unit radius. Each of them is initialized by either a mapped Fibonacci lattice [Purser 2008] or random uniform points depending on whether it is a spherical or intra-spherical component.

Then, the radii of each layers is determined by:

$$r_i = \frac{2^{i-1}}{2^8 \times R\left(\hat{Q}_i\right)} \lambda_i, \tag{2}$$

where 2^8 is the resolution of first mipmap, λ_i is an empirical parameter for controlling overlapping ratio of the stencil layers, and we suggest $\lambda_i = 1.0 - 0.08 \times \min(i, 5)$. Finally, Q_i is obtained through scaling each component of $\hat{Q}_i$ by r_i.

The high-frequency part. The high-frequency part of the stencil points are trivially spaced along the directional light vector. Recall that the mip-level is higher as a point moves away from the center (i.e. the corresponding volume is larger), the non-overlapping points form a cone, which is similar to the pattern in cone tracing [Crassin et al. 2011]. It consists of 4 layers (indexed 9 to 12), each of which contains 8 points, and therefore totally 32 points in the high-frequency part. The mip-level of a layer Q_i in this part is $m_i = i - 9$ analogously.

A.3 Descriptor Sampling

We assume the density outside the volume boundary is 0. However, extra attention should be paid to the scaled-transmittance fields, where hardware clamping could result in incorrect samples, as illustrated in Fig . 3(a). We redirect the sample points outside the volume to the intersection with the volume's boundary in the direction of light for the accurate values (Fig. 3(b)).

A.4 Cumulative Phase Function

We integrate the phase function within an angle in each direction to get the cumulative phase function:

$$p'(\omega, \theta, c) = \int_{\Omega} p(\omega, \phi) d\phi \ , \ \Omega = \{\phi | \ (\theta \cdot \phi) > \cos(c/2)\} . \tag{3}$$

Here, p is the original phase function, and angle c represents the projection of the volume from the current stencil point onto the solid angle at point $\mathbf{u}$.

A.5 Architectures of MRPNN

Before going over the two stages of our network, we first present the SE (Squeeze-and-Excitation) module and a feature block.

SE module. The SE module consists of a squeeze phase and an excitation phase.

In the *squeeze* phase of SE module, we apply average and max pooling to the input feature $\boldsymbol{F}_i^c \in \Sigma_i, c \in \{\mu, S, P\}$, and then use the extracted information and the input parameters $\{G, \gamma\}$ to construct an 8D vector $\boldsymbol{e}_i$:

$$\begin{aligned} \bar{t}_i^c &= \frac{1}{N_i} \sum_{j=1}^{N_i} \boldsymbol{F}_{i,j}^c, \\ \tilde{t}_i^c &= \max \boldsymbol{F}_{i,j}^c, \\ \boldsymbol{e}_i &= \{\bar{t}_i^\mu, \bar{t}_i^S, \bar{t}_i^P, \tilde{t}_i^\mu, \tilde{t}_i^S, \tilde{t}_i^P, G, \gamma\}, \end{aligned} \tag{4}$$

where $j \in \mathbb{Z}_{[1,N_i]}$, N_i corresponds to the number of points in the i-th layer of stencil, and $\boldsymbol{F}_{i,j}^c$ is the j^{th} feature in $\boldsymbol{F}_i^c$.

In the *excitation* phase of SE module, we compute the weights of the input features $\boldsymbol{F}_i^c$ as :

$$\boldsymbol{w}_i = SE_{ex}(\boldsymbol{e}_i) = \{w_i^\mu, w_i^S, w_i^P\} = \varsigma(\boldsymbol{V}_{i,2}\delta(\boldsymbol{V}_{i,1}\boldsymbol{e}_i)), \tag{5}$$

where $\boldsymbol{V}_{i,1}, \boldsymbol{V}_{i,2}$ are trainable weights, $\delta(x) = \max(x, 0)$ denotes the ReLU activation function, $\varsigma(x) = \frac{1}{1+e^{-x}}$ denotes the Sigmoid activation function, and SE_{ex} denotes the excitation phase of the SE module.

Then our full SE module structure scales $\boldsymbol{F}_i^c$ as :

$$\begin{aligned} w_i^c &\in SE_{ex}(\boldsymbol{e}_i), \\ \hat{\boldsymbol{F}}_i^c &= SE(\boldsymbol{F}_i^c) = w_i^c \cdot \boldsymbol{F}_i^c, \end{aligned} \tag{6}$$

where SE denotes the full Squeeze-and-Excitation module.

Feature block. In the feature block, the input will be processed by SE module first. After that, we apply a fully connected layer and activation function to the scaled features to compute the output of feature block $Block_F$:

$$\boldsymbol{o}_i^c = Block_F(\boldsymbol{F}_i^c, \boldsymbol{o}_{i-1}^c) = \delta\left(FC_{N_i}^{N_i}\left(\boldsymbol{o}_{i-1}^c + SE(\boldsymbol{F}_i^c)\right)\right) + \boldsymbol{F}_i^c, \tag{7}$$

where $\boldsymbol{o}_{i-1}^c$ is the output of the former feature block [1] [2] , $FC_{out}^{in}(X) = \boldsymbol{W}X + \boldsymbol{b}$ denotes the fully connected layer, $\boldsymbol{W}, \boldsymbol{b}$ are trainable weights and biases in FC, and $\{in, out\}$ are the numbers of input and output channels in FC.

Feature stage. As shown in Fig. 2, in *feature* stage, low-frequency sub-network uses $M = 8$ feature blocks to fuse the low-frequency part of stencil, and high-frequency sub-network uses $K - M = 4$ feature blocks to fuse the high-frequency part.

Then, we apply a single fully connected layer as the last layer of low- and high-frequency sub-network to output the final latent

[1]When $i = 0$ or $i = M + 1$, we let $\boldsymbol{o}_{i-1}^c = \boldsymbol{0}$.
[2]When $N_{i-1} \neq N_i$, we use concatenation instead of addition, and Eq. 7 turns into $\boldsymbol{o}_i^c = \delta\left(FC(\boldsymbol{o}_{i-1}^c + SE(\boldsymbol{F}_i^c))\right) \| \boldsymbol{F}_i^c$.

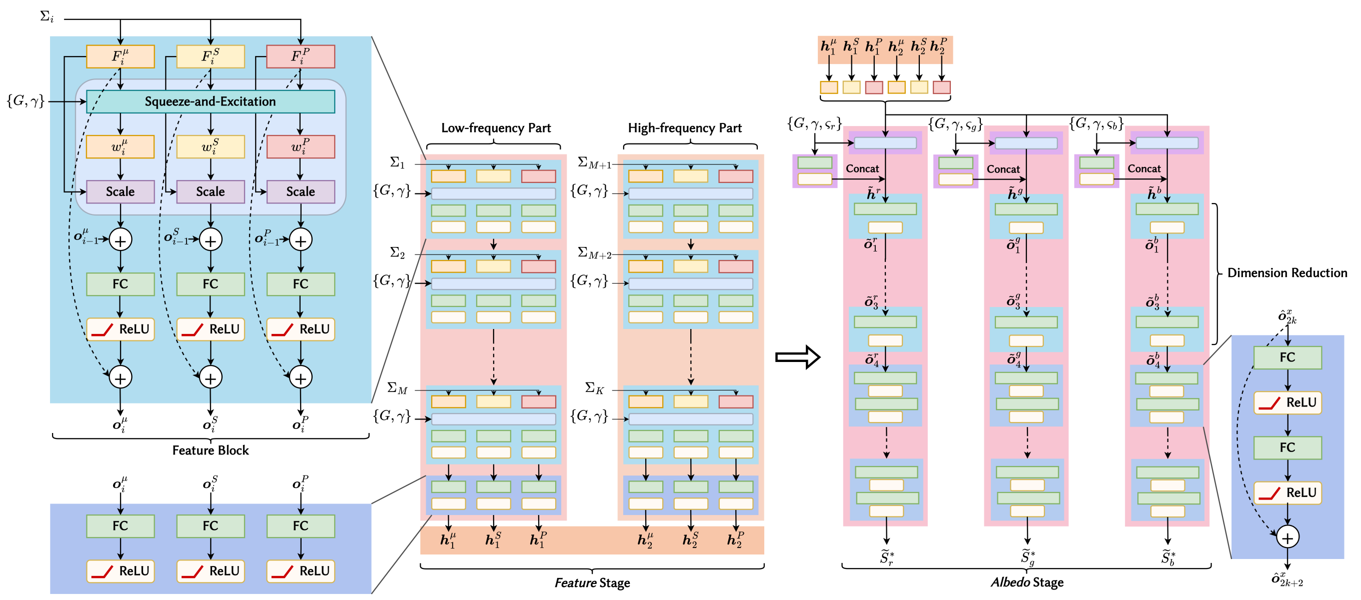


Figure 2: An overview of the full network structure. FC denotes a fully-connected layer.

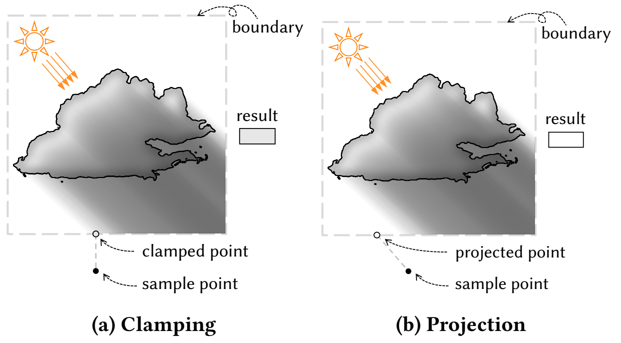


Figure 3: The sampled transmittance values are incorrect with hardware clamping, which is solved by projecting them onto the volume's boundary.

vectors:

$$h_1^c = \delta\left(FC_{N_M}^{N_M}(o_M^c)\right), \quad (8)$$

$$h_2^c = \delta\left(FC_{N_K}^{N_K}(o_K^c)\right), \quad (9)$$

where o_M^c, o_K^c are the output of the final feature block of low- and high-frequency sub-network, h_1^c is a 32D low-frequency vector, and h_2^c is an 8D high-frequency vector.

Albedo stage. In the *albedo* stage, we input $h_1^c, h_2^c, \{G, \varsigma_x, \gamma\}$ into three sub-networks to fuse the albedo features for each spectral channel $x \in \{r, g, b\}$.

First, we introduce albedo $\varsigma_{x,x\in\{r,g,b\}}$ to input parameters $\{G, \gamma\}$ as $h^{\varsigma_x} = \{G, \varsigma_x, \gamma\}$, then get an 8D albedo-dependent latent vector as:

$$\hat{h}^{\varsigma_x} = \delta\left(FC_8^3(h^{\varsigma_x})\right). \quad (10)$$

Then, a SE Module is performed for fusing h_1^c, h_2^c with the albedo of each spectral channel. In the squeeze phase, we first construct a 75D vector for each spectral channel x by concatenating the intermediate value produced by pooling:

$$\hat{e}^{\varsigma_x} = \{\bar{t}_i^\mu, \bar{t}_i^S, \bar{t}_i^P, \tilde{t}_i^\mu, \tilde{t}_i^S, \tilde{t}_i^P, \}_{i=1}^K \cup \{\varsigma_x, G, \gamma\}, \quad (11)$$

where ς_x is the albedo of the spectral channel x.

The latent vectors h_1^c, h_2^c from *feature* stage are then scaled by:

$$\begin{aligned} \hat{w}_f^{cx} &\in \{\hat{w}_1^{\mu x}, \hat{w}_1^{Sx}, \hat{w}_1^{Px}, \hat{w}_2^{\mu x}, \hat{w}_2^{Sx}, \hat{w}_2^{Px}\} = SE_{ex}(\hat{e}^{\varsigma_x}), \\ \hat{h}_f^{cx} &= \hat{w}_f^{cx} \cdot h_f^c, \end{aligned} \quad (12)$$

where $c \in \{\mu, S, P\}$ are the feature channels, $f \in \{1, 2\}$ are the low- and high- features, and SE_{ex} denotes the excitation phase of SE module.

We then concatenate the previous vector $\hat{h}^{\varsigma_x}$ with the scaled feature $\hat{h}_f^{cx}$ through $\tilde{h}^x = \{\hat{h}_f^{cx}\}_{c\in\{\mu,S,P\},f\in\{1,2\}} \cup \{\hat{h}^{\varsigma_x}\}$ for network inference, and $\tilde{h}^x$ is a 128D vector.

Next, we use $\tilde{h}^x$ to process the final output. To accelerate the process, we first use four fully connected layers with ReLU activation function to perform a dimensional reduction, the input channel of i^{th} fully connected layer [3] is $128/2^{i-2}$, the output channel is $128/2^{i-1}$,and a 16-D vector $\tilde{o}^x$ is the resulting of dimensional reduction.

Finally, $\tilde{o}^x$ is sent into $H = 12$ fully connected layers with residual connection:

$$\begin{aligned} \hat{o}_{2k+1}^x &= \delta\left(FC_{16}^{16}(\hat{o}_{2k}^x)\right), \\ \hat{o}_{2k+2}^x &= \delta\left(FC_{16}^{16}(\hat{o}_{2k+1}^x)\right) + \hat{o}_{2k+1}^x, \end{aligned} \quad (13)$$

[3]Except for $i = 1$, where input channel is 128 and output channel is 128.

where $\hat{\boldsymbol{o}}_0^x = \tilde{\boldsymbol{o}}^x$ is the input, $k \in \{0, 1, ..., H/2 - 1\}$, and those fully connected layers [4] output a scalar $\widetilde{S}_x^*$.

The inference executes for each $\varsigma_{x, x\in\{r,g,b\}}$ to compose the in-scattering radiance $\{\widetilde{S}_r^*, \widetilde{S}_g^*, \widetilde{S}_b^*\}$.

A.6 Architectures of MRPNN-Narrow and MRPNN-Wide

The narrow variation. We remove the SE Module (i.e., erasing all connections between features throughout the *feature* stage), such that its subnetwork structure in the *feature* stage is:

$$\boldsymbol{o}_i^c = \delta\left(FC_{N_i}^{N_i}(\boldsymbol{o}_{i-1}^c + \boldsymbol{F}_i^c)\right) + \boldsymbol{F}_i^c, \tag{14}$$

and $\tilde{\boldsymbol{h}}^x$ is established by concatenating the results in the *albedo* stage:

$$\tilde{\boldsymbol{h}}^x = \{\boldsymbol{h}_1^\mu, \boldsymbol{h}_1^S, \boldsymbol{h}_1^P, \boldsymbol{h}_2^\mu, \boldsymbol{h}_2^S, \boldsymbol{h}_2^P, \hat{\boldsymbol{h}}^{\varsigma_x}\}. \tag{15}$$

The wide variation. To fuse the features, the wide variation applies fully-connected layers on the concatenation of the features:

$$\boldsymbol{F}_i = \{\boldsymbol{F}_{i,1}^\mu, ..., \boldsymbol{F}_{i,N_i}^\mu, \boldsymbol{F}_{i,1}^S, ..., \boldsymbol{F}_{i,N_i}^S, \boldsymbol{F}_{i,1}^P, ..., \boldsymbol{F}_{i,N_i}^P\}, \tag{16}$$

where the sub-network structure is:

$$\boldsymbol{o}_i = \delta\left(FC_{3N_i}^{3N_i}(\boldsymbol{o}_{i-1} + \boldsymbol{F}_i)\right) + \boldsymbol{F}_i. \tag{17}$$

The last layer that outputs the latent vectors in the low-frequency sub-network is:

$$\boldsymbol{h}_1 = \delta\left(FC_{96}^{96}(\boldsymbol{o}_M)\right), \tag{18}$$

and that in the high-frequency sub-network is:

$$\boldsymbol{h}_2 = \delta\left(FC_{24}^{24}(\boldsymbol{o}_K)\right), \tag{19}$$

respectively.

Finally, in the *albedo* stage, the concatenated vector $\tilde{\boldsymbol{h}}^x$ is:

$$\tilde{\boldsymbol{h}}^x = \{\boldsymbol{h}_1, \boldsymbol{h}_2, \hat{\boldsymbol{h}}^{\varsigma_x}\}. \tag{20}$$

Note that the scale of the entire network is considerably larger than our design. This accounts for the slower convergence and performance.

REFERENCES

Cyril Crassin, Fabrice Neyret, Miguel Sainz, Simon Green, and Elmar Eisemann. 2011. Interactive indirect illumination using voxel cone tracing. In *Computer Graphics Forum*, Vol. 30. Wiley Online Library, 1921–1930.

Stefan Heinrich, Alexander Keller, and Harald Niederreiter. 2008. *(t, m, s)-Nets and Maximized Minimum Distance.* Springer, 397–412.

Robert James Purser. 2008. Generalized fibonacci grids; a new class of structured, smoothly adaptive multi-dimensional computational lattices. (2008).

[4] To output the scalar, for $k = H/2 - 1$, the output channel of the $2k + 2^{th}$ fully connected layer is set to 1.