Fast Blue-Noise Generation via Unsupervised Learning

Daniele Giunchi[§]
Computer Science Department
University College London
London, United Kingdom
d.giunchi@ucl.ac.uk

Alejandro Sztrajman[§]
Computer Science Department
University College London
London, United Kingdom
a.sztrajman@ucl.ac.uk

Anthony Steed
Computer Science Department
University College London
London, United Kingdom
a.steed@cs.ucl.ac.uk

***Abstract*—Blue noise is known for its uniformity in the spatial domain, avoiding the appearance of structures such as voids and clusters. Because of this characteristic, it has been adopted in a wide range of visual computing applications, such as image dithering, rendering and visualisation. This has motivated the development of a variety of generative methods for blue noise, with different trade-offs in terms of accuracy and computational performance. We propose a novel unsupervised learning approach that leverages a neural network architecture to generate blue noise masks with high accuracy and real-time performance, starting from a white noise input. We train our model by combining three unsupervised losses that work by conditioning the Fourier spectrum and intensity histogram of noise masks predicted by the network. We evaluate our method by leveraging the generated noise for two applications: grayscale blue noise masks for image dithering, and blue noise samples for Monte Carlo integration.**

***Index Terms*—Neural networks, Image processing, Noise generators, Monte Carlo methods**

I. Introduction

Noise signals are commonly classified by the characteristics of their Fourier spectra, following a colour-based naming convention. In analogy with white light, white noise corresponds to a flat spectrum, with similar intensities for all frequencies. Conversely, blue noise refers to a spectrum with minimal low-frequency components and no high-intensity spikes [2]. Intuitively, blue-noise sampling gives place to a randomised spatially-uniform distribution of samples, playing an important role in many applications within visual computing, such as rendering [3]–[5], dithering [6], stippling [7], [8], texture synthesis [9], simulations [10] and data visualisation [13].

The generation of blue-noise masks is a computationally demanding process, and a large number of methods have been developed for this purpose, with different trade-offs in terms of speed and spectral accuracy [11]. In this work, we present a new method for blue-noise mask generation based on the unsupervised training of a neural network, that can produce blue-noise masks at real-time rates. To our knowledge, this is the first method to use a neural-based approach for the critical task of blue-noise generation. In contrast with other methods, our approach can be easily generalised to any kind of noise spectrum, and the inference time does not increase considerably with the texture size. Our method relies on the combination of three unsupervised losses that together constrain the predicted noise to fit the mathematical definition of blue noise (described in mathematical detail in Section III). The first two losses act on the Fourier spectrum of the noise signal, penalising high frequencies and the occurrence of high-intensity peaks, while the third loss forces the histogram of intensities to remain uniform. In Section IV-A, we show that our predicted blue-noise masks can be used for image dithering, and we compare our results with other approaches in terms of dithering error and inference time. Finally, in Section IV-B we convert our grayscale blue-noise masks to binary masks via thresholding, obtaining blue-noise samples that we leverage for Monte-Carlo integration.

[§]Equal contribution

II. Related Work

A. Blue Noise

Blue noise is generally characterised by a reduced set of low frequencies, absence of peaks in high frequencies [1]. This is a property found in the spatial organisation of retinal cells, and thus it has been linked with the production of perceptually pleasing patterns [22]. Consequently, it has been widely adopted in many visual computing applications [2], including dithering and rendering, which we will discuss in Sections II-B and II-C. A constant density of samples characterises a blue noise distribution at different scales. Geometrically speaking, features such as voids or clusters are not present. This geometric property appears in the Fourier domain as the uniform presence of high frequencies and the absence of dominant components and low frequencies.

There are multiple ways to characterise blue noise patterns, e.g., according to their dimensionality, the style of the generating algorithm (dart throwing, relaxation or tiling), and whether the output is a blue noise grayscale mask or a set of point samples [2]. So far, most methods for blue noise have focused on the generation of sample sets [15]–[17]. In contrast, only a few methods exist to generate blue-noise masks, which are required for blue-noise dithering. Moreover, blue-noise masks can be leveraged for the generation of sample sets, through thresholded binarisation, offering more flexibility and practical applications [2], [5].

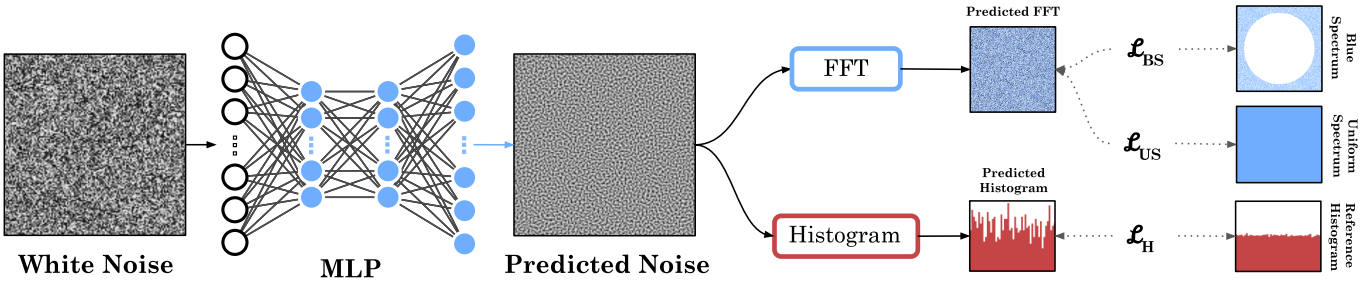

Fig. 1. Diagram of network training, combining the three unsupervised losses: uniform spectrum loss $\mathcal{L}_{\text{US}}$, blue spectrum loss $\mathcal{L}_{\text{BS}}$, and histogram loss $\mathcal{L}_{\text{H}}$. After training, the network produces blue noise grayscale masks based on white noise input.

B. Dithering

Dithering can be defined as the intentional addition of noise to a signal, in order to prevent the occurrence of reconstruction errors due to quantisation or undersampling. Methods for dithering can be applied in different signals such as audio and images, and enable data compression while preserving the quality of the signal [20]. A common application of dithering on image data is halftoning, i.e., displaying images as monochromatic or using one bit per channel. Dithering techniques exploit properties in the spatial integration produced in the human visual system by distributing the errors among pixels, thus reducing the quantisation effects [12].

The most prevalent method for blue noise mask generation for dithering is the void and cluster algorithm by Ulichney [18]. The method works by filling the voids between pixels until eliminating empty regions. Although this procedure is computationally expensive, especially for high-resolution masks, it is able to generate high-quality blue noise masks in any dimensionality. More recently, Wronski [14] developed an optimisation-based method that leverages three losses to generate blue-noise masks from white noise masks. Our method, described in Section III, uses similar losses to produce blue masks, however we leverage a neural network architecture to reduce the inference time of the noise.

C. Blue Noise and Monte Carlo Integration

Blue noise grayscale masks can be used to produce blue noise samples, through the application of a thresholded binarization [2], [5]. This enables further applications for blue noise, such as Monte Carlo integration, which we will discuss here. Monte Carlo integration (MCI) is a mathematical approach that evaluates integrals by using random numbers or samples [24], [25], [29]. Such an approach can be achieved via different techniques: uniform sampling [26], importance sampling [27], stratified sampling [28], particle filter [29] or mean-field particle techniques [30]. One application of MCI is in rendering; in fact, rendering methods usually involve the numerical computation of high-dimensional integrals. This is typically done via sampling algorithms, and hence the generation of samples is a core problem of the area [2]. The application of blue noise patterns to MCI for rendering was made popular by Mitchell [21], and has received significant attention since then. Due to its low discrepancy and randomness, blue-noise sampling has been leveraged to increase rendering quality and efficiency. Spencer and Jones [19] applied blue-noise sampling to the noise reduction in renderings of caustics via photon mapping. Georgiev and Fajardo [5] exploited blue noise masks for Monte Carlo ray tracing, demonstrating impressive results in terms of noise reduction for low samples per pixel.

III. Method

Our approach for blue noise generation is depicted in Figure 1, where we show a diagram of our network training scheme. The architecture of the network is composed of simple fully-connected layers with hyperbolic tangent activations, using noise masks of e.g. 128×128 as input and output (thus the dimensions of the network layers are $16384 \times 100 \times 100 \times 16384$).

For the training phase, we leverage a linear combination of three unsupervised losses that enforce mathematical constraints on the output noise:

$$\mathcal{L}_{\text{US}}\text{: Uniform spectrum loss}$$
$$\mathcal{L}_{\text{H}}\text{: Histogram loss}$$
$$\mathcal{L}_{\text{BS}}\text{: Blue spectrum loss}$$

The $\mathcal{L}_{\text{US}}$ and $\mathcal{L}_{\text{BS}}$ losses operate on the Fourier spectrum of the output noise. The `uniform spectrum loss` $\mathcal{L}_{\text{US}}$ imposes a penalty on the magnitude of the directional derivatives of the Fourier spectrum, thus favouring a homogeneous spectrum of energies and preventing the appearance of high-intensity peaks. The `blue spectrum loss` $\mathcal{L}_{\text{BS}}$ penalises large values of the Fourier spectrum on a central circular region corresponding to the low frequencies. In combination, these two losses produce a Fourier spectrum such as observed in Figure 2 on the left. In contrast, the `histogram loss` $\mathcal{L}_{\text{H}}$ operates on image-space, enforcing a uniform distribution of intensity levels in Figure 2 on the right. Such loss functions map the objective function of the [14] method. We adopt the same weighted coefficients in the linear combination of the losses and then, train our weights.

We ran our training with 32k samples for 12 epochs, using the Adam optimiser, a learning rate of $5e^{-4}$, and a batch size of 64. For the input and output, we tested resolutions from

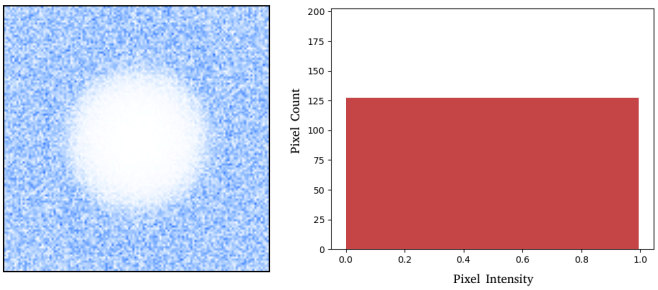

Fig. 2. Left: blue noise Fourier spectrum enforced by the $\mathcal{L}_{\text{US}}$ and $\mathcal{L}_{\text{BS}}$ losses. Right: uniform intensity histogram enforced by the $\mathcal{L}_{\text{H}}$ loss.

64×64 to 512×512 pixels, with a maximum training time of $120s$ on a GeForce GTX 1080 Ti. The code was implemented using Tensorflow 2.7 in Python 3, and can be found in the project's webpage.

A. Mathematical Description of Loss Functions

The total loss function is a linear combination of three separate components described in the previous section: $\mathcal{L}_{\text{US}}$, $\mathcal{L}_{\text{H}}$ and $\mathcal{L}_{\text{BS}}$. The blue spectrum loss $\mathcal{L}_{\text{BS}}$ acts on the Fourier spectrum of the predicted noise, penalising frequencies increasingly above a cut-off threshold ω_{cutoff}. Let $\hat{s}$ be the noise signal predicted by the network, and $\hat{\phi}_{ij} = \|\text{FFT}(\hat{s})\|$ the norm of its Fourier spectrum at grid point ij. Then we can write the blue spectrum loss as:

$$\mathcal{L}_{\text{BS}} = \sum_{ij} \hat{\phi}_{ij} \left[\max\left(0, \frac{\omega_{\text{cutoff}} - r_{ij}^2}{\omega_{\text{cutoff}}}\right)\right]^2 \tag{1}$$

where r_{ij} is the radius of the point ij measured from the center of the grid.

The uniform spectrum loss $\mathcal{L}_{\text{US}}$ also acts on the spectrum of the predicted signal $\hat{s}$, by penalising large values of its spatial derivatives:

$$\mathcal{L}_{\text{US}} = \sum_{ij} \left[\nabla^2 \hat{\phi}_{ij}\right]^2 + \nabla_i \hat{\phi}_{ij} + \nabla_j \hat{\phi}_{ij} \tag{2}$$

where ∇^2 is the Laplacian operator, and ∇_i and ∇_j are the spatial derivatives in the main directions of the grid, and can be computed by finite differences.

Finally, the histogram loss $\mathcal{L}_{\text{H}}$ acts on the histogram of intensities of the noise:

$$\mathcal{L}_{\text{H}} = \sum_{k} \left[\text{Histogram}(\hat{s})_k - \text{RefH}_k\right]^2 \tag{3}$$

by penalising deviations from a uniform reference histogram RefH.

IV. Results

A. Dithering

After training, our network is able to produce blue noise masks as output, as shown in the first column of Figure 3. Here we display typical blue noise masks for multiple noise generation methods. In addition, to regular white noise, we show outputs from the popular Void & Cluster method by Ulichney [18], and the optimisation-based approach proposed by Wronski [14]. In columns 2 and 3 of Figure 3 we display diagrams of the histogram and Fourier spectra of the predicted masks, where we can observe the effects of the unsupervised losses described in Section III.

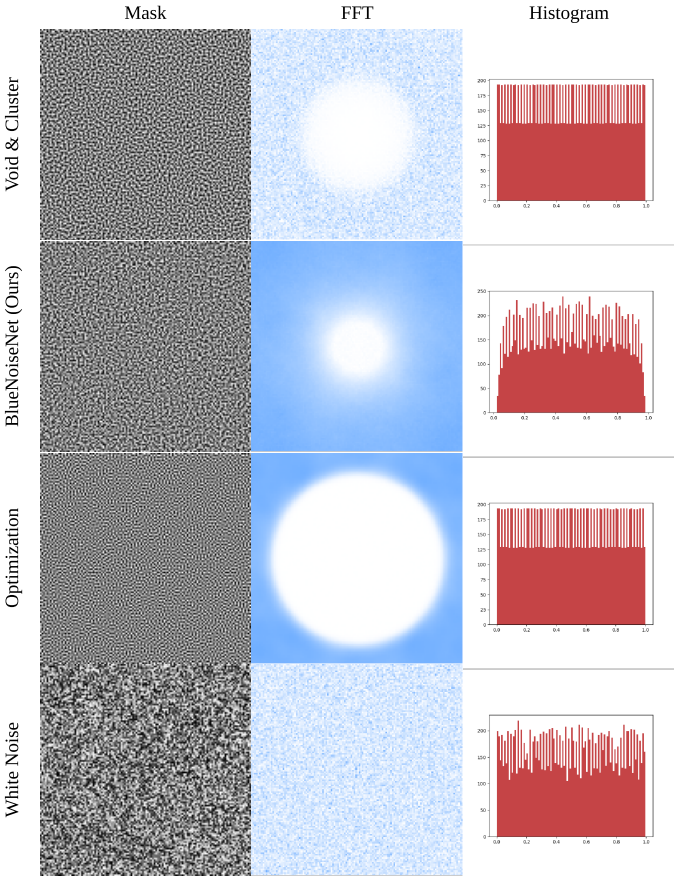

Fig. 3. Comparison between noise masks, FFT of such mask, and histogram. On the rows different method to obtain such mask.

In order to evaluate our results, we will focus on the application of blue-noise to image dithering, discussed in Section II-B. In Figure 4 we display a comparison of the dithering of multiple images, using the different methods from Figure 3. In all cases, dithering is used to compress the images to a single bit per colour channel.

A visual inspection of Figure 4 shows that our method produces dithered reconstructions with lower noise than white noise, and similar in quality to the optimisation-based approach by Wronski [14]. Consistently, the Void & Cluster method produces the highest quality reconstructions, although we will later show that this is at the expense of a high computational cost, which might be prohibitive for real-time applications. The observations from Figure 4 are confirmed by Table I, where we show average root mean square errors (RMSE) for the dithering of the David statue, using 100

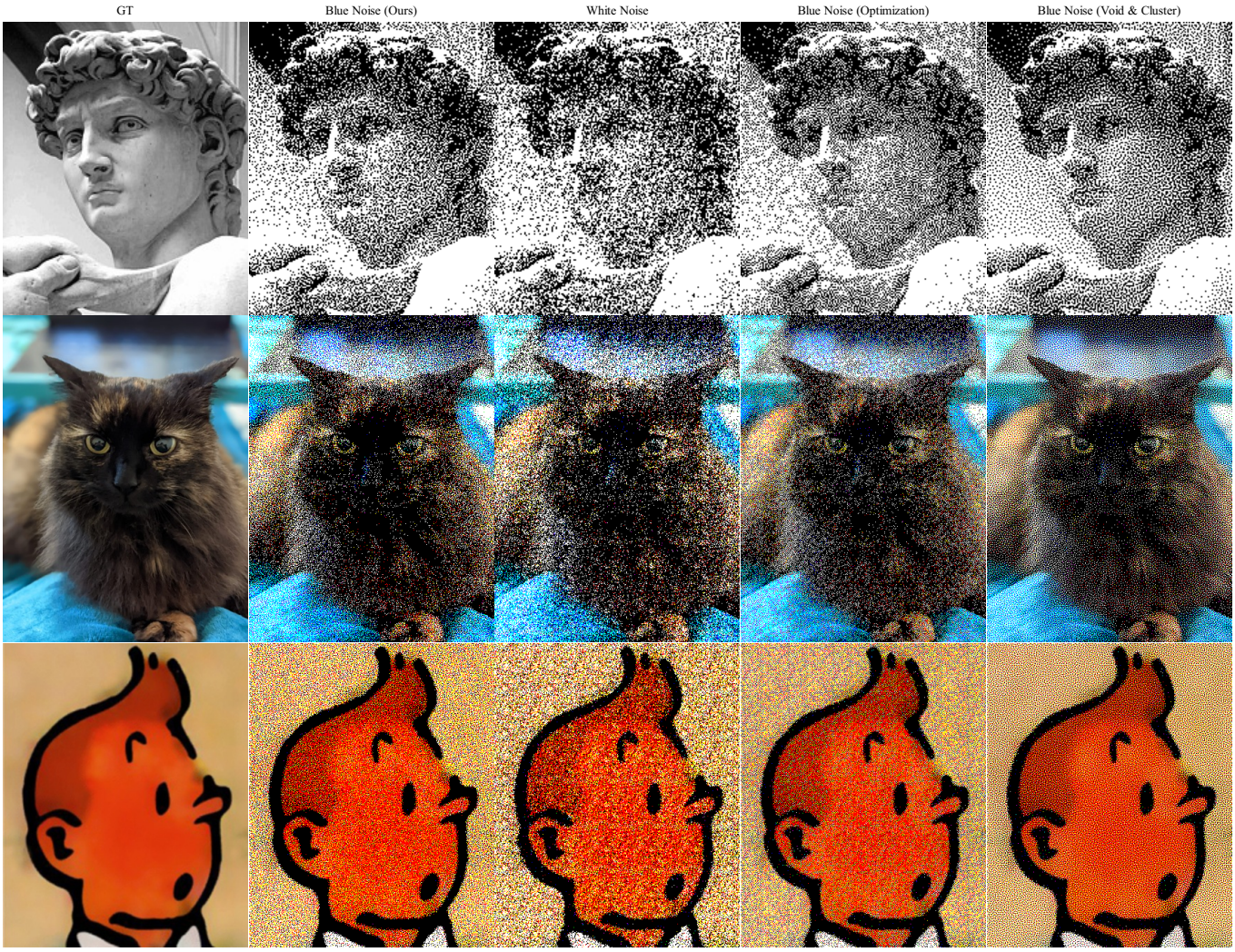

Fig. 4. Comparison between dithering results, using our solution (second column) and others such as white noise, Wronski 2020, and Void&Cluster. The left most column is the original image.

different dithering masks for each noise generation method. As part of the error computation, both the ground truth and dithered images are pre-processed with a Gaussian filter.

TABLE I
DITHERING ERRORS FOR MULTIPLE NOISE GENERATION METHODS, AVERAGED OVER 100 DITHERING MASKS.

Method	RMSE
White Noise	0.069 ± 0.002
Neural Network (Ours)	0.046 ± 0.001
Wronski 2020 [14]	0.046 ± 0.001
Void & Cluster	0.029 ± 0.001

Finally, we analyse in Figure 6 the inference time (in logarithmic scale) of the different noise generation methods, as a function of the dither mask resolution. Here we see that our method is able to produce new blue noise masks much faster than other methods, at around 0.018 s per mask. Moreover, the evaluation time for other methods grows considerably with the size of the noise mask, while our method's remains nearly constant. This opens the door for interesting applications, such as fast blue noise sampling for Monte Carlo based rendering, as discussed in Section II-C.

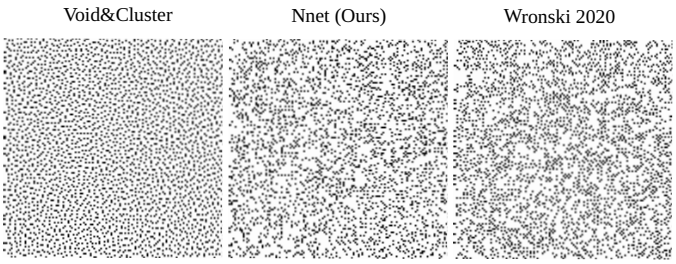

Fig. 5. Samples obtained by thresholded binarization of the blue noise grayscale masks produced by three different methods.

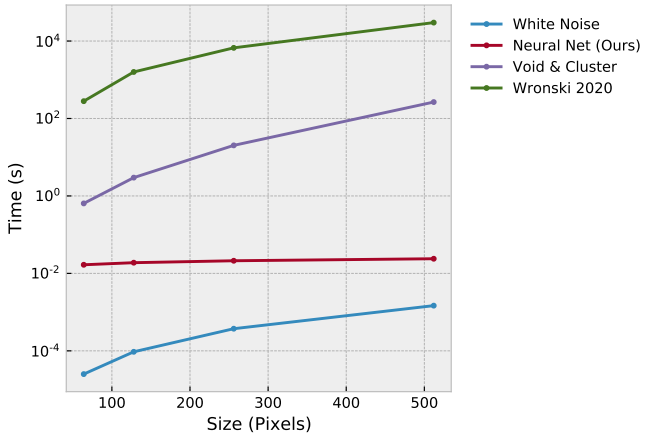

Fig. 6. Time (log) vs mask size for different noise generation methods.

B. Monte Carlo Integration

As discussed in Section II-C, blue noise masks can also be leveraged to produce sample sets, enabling applications of blue noise sampling, such as efficient Monte Carlo integration. In Figure 5 we display exemplars of blue noise samples with multiple noise generation methods analysed in the previous section.

We evaluate the blue noise samples generated by these methods through the application of Monte Carlo integration. In Figure 7 we leverage the generated samples to the integration of two functions, with known analytic integrals. We show the root mean square error of the integral computation through Monte Carlo sampling, as a function of the number of samples used for the integration.

As expected, white noise and void & cluster present correspondingly the slowest and fastest convergence speeds, in terms of the number of samples. As seen in the previous section, in the case of void & cluster this comes at the cost of a high inference time of the blue noise mask, especially for large mask sizes, potentially required to generate large numbers of samples. Our neural-based method shows consistently faster convergence than white noise, and in some cases lower error than void & cluster, for low number of samples. The method by Wronski [14] presents an inconsistent behaviour, showing in some cases even higher errors than white noise integration. It is worth noting that, while other potentially faster methods exist for the generation of low-discrepancy samples, we have reduced our comparison to methods that are able to produce blue noise masks for dithering.

V. Limitations and Future Work

The implementation of the method for blue noise generation described in Section III is limited to bi-dimensional noise masks. However, there is no intrinsic limitation to extend the approach to higher dimensionality. Efficiently producing

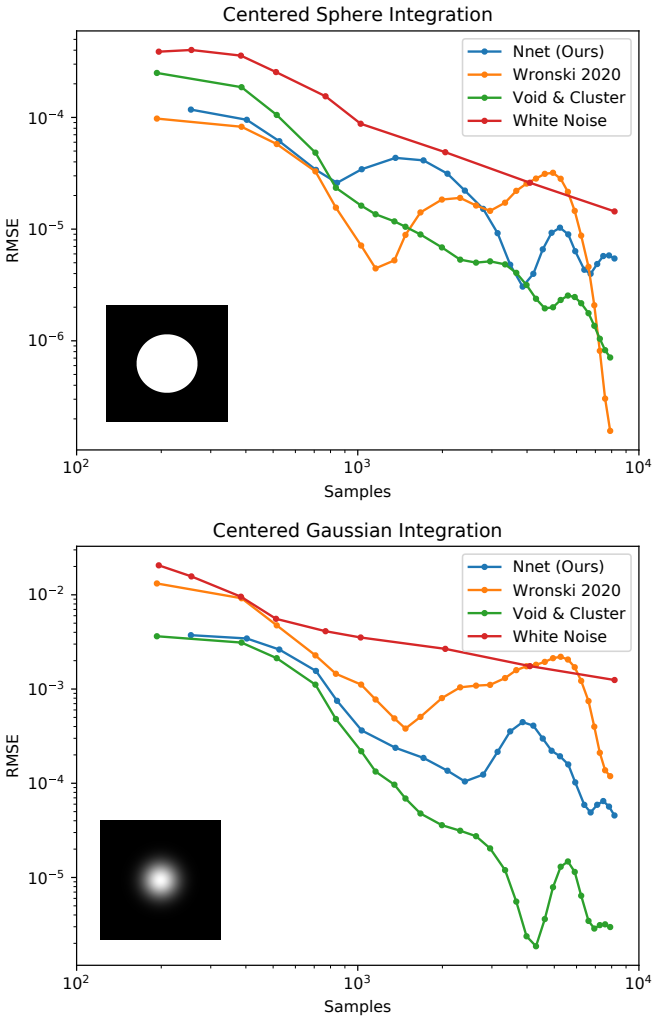

Fig. 7. RMSE vs Number of Samples (log-log scale) for the Monte Carlo integration of two functions (rendered in the bottom left corners), using samples from different noise generation methods. Top: Centred Sphere. Bottom: Centred Gaussian. [31]

higher-dimensional blue noise is a current open problem that would benefit Monte Carlo integration for ray-tracing. The complexity of methods such as void & cluster grows intractably with the dimensionality of the noise.

On the other hand, for bi-dimensional cases, we have shown that the void & clusters method produces very high quality blue-noise, very hard to match by other algorithms. Following our current approach, an interesting direction for research would be to leverage the void and cluster algorithm in the loss as a differentiable operation. Another strong candidate to improve the loss in our architecture, is the inclusion of a dithering-based loss, i.e., a differential implementation of image dithering that can improve noise generation specifically for dithering. In addition, other geometric losses can be incorporated, such as the star-discrepancy loss, that takes into account the average distance between points. Other potential improvements to the architecture would involve splitting the

Fourier spectrum into real and imaginary parts inside the network architecture, to operate on them separately, and the inclusion of an adversarial loss that leverages blue noise from the void & cluster method.

VI. Conclusions

In this work, we present a new method for blue-noise mask generation based on the unsupervised training of a neural network. Our method can produce blue-noise mask from white-noise inputs at interactive rates. To our knowledge, this is the first method to use a neural-based approach to generate blue-noise masks, which can be leveraged for image dithering. Our approach can be easily generalised to any kind of noise spectrum, and we observed that the inference time does not increase considerably with the output mask size.

We compared our method against other blue-noise generation techniques that are able to produce grayscale masks. We evaluated all methods by measuring the average errors on two applications of blue noise: image dithering and Monte Carlo integration. While the void & cluster method produces the most accurate blue-noise, we showed that our method is faster, especially for large mask sizes.

Acknowledgements

This work has been supported by H2020 EU project RISE, grant agreement No 739578. The authors would like to thank Stephanie Urquhart for providing the picture of *Meadow* for Figure 4.

References

[1] Heck, Daniel, Thomas Schlömer, and Oliver Deussen. "Aliasing-Free Blue Noise Sampling." (2012).

[2] D. M. Yan, J. W. Guo, B. Wang, X. P. Zhang and P. Wonka. A Survey of Blue-Noise Sampling and Its Applications. J. Comput. Sci. Technol. 30, pp. 439–452 (2015).

[3] D. P. Mitchell. Generating antialiased images at low sampling densities. Proc. ACM SIGGRAPH, pp. 65–72 (1987).

[4] G. Singh, C. Oztireli, A. G. Ahmed, D. Coeurjolly, K. Subr et al. Analysis of sample correlations for Monte Carlo rendering. Comp. Graph. Forum (Proc. EGSR) 38, 2 (2019).

[5] I. Georgiev and Marcos Fajardo. 2016. Blue-noise dithered sampling. In ACM SIGGRAPH 2016 Talks. Association for Computing Machinery, New York, NY, USA, Article 35, 1.

[6] R. A. Ulichney. Dithering with blue noise. Proc. IEEE 76, 1 (1988).

[7] A. Secord. Weighted Voronoi stippling. In Proc. NPAR 2002. pp. 37-43 (2002).

[8] R. Fattal. Blue-noise point sampling using kernel density model. ACM Trans. on Graph. (Proc. SIGGRAPH). 28 (3), pp. 48:1–48:10 (2011).

[9] F. Wu, W. Dong, Y. Kong, X. Mei, D.-M. Yan et al. Feature-aware natural texture synthesis. The Visual Computer, pp. 1–13 (2014).

[10] H. Schechter and R. Bridson, "Ghost SPH for animating water," ACM Trans. Graph. (Proc. SIGGRAPH). 31 (4), pp. 61:1–61:8 (2012).

[11] N. K. Kalantari and P. Sen. Fast Generation of Approximate Blue Noise Point Sets. Comp. Graph. Forum. 31 (4), pp. 1529-1535 (2012).

[12] L. Yue, P. Ganesan, B. S. Sathish, C. Manikandan, A. Niranjan et al. The Importance of Dithering Technique Revisited With Biomedical Images—A Survey. IEEE New Trends in Brain Signal Processing and Analysis. vol. 7, pp. 3627-3634 (2019).

[13] C. van Onzenoodt, G. Singh, T. Ropinski and T. Ritschel. Blue Noise Plots. Comp. Graph. Forum, vol. 40, no. 2, pp. 425-433 (2021).

[14] B. Wronski. Optimizing blue noise dithering: backpropagation through Fourier transform and sorting. Retrieved from http://bartwronski.com. Last access 23rd January 2022.

[15] Dunbar D, Humphreys G. A spatial data structure for fast Poisson-disk sample generation. ACM Trans. Graphics, 2006, 25(3): 503-508.

[16] White K B, Cline D, Egbert P K. Poisson disk point sets by hierarchical dart throwing. In Proc. IEEE Symposium on Interactive Ray Tracing, Sept. 2007, pp.129-132.

[17] Gamito M N, Maddock S C. Accurate multidimensional Poisson-disk sampling. ACM Trans. Graphics, 2009, 29(1): 8:1-8:19

[18] Robert A. Ulichney "Void-and-cluster method for dither array generation", Proc. SPIE 1913, Human Vision, Visual Processing, and Digital Display IV, (8 September 1993)

[19] B. Spencer and M. W. Jones. Progressive photon relaxation. ACM Transactions on Graphics, 32 (1) 7:1-7:11 (2013).

[20] A. Wolfe, N. Morrical, T. Akenine-Möller and R. Ramamoorthi, Scalar Spatiotemporal Blue Noise Masks, arXiv:2112.09629.

[21] Don P. Mitchell. 1991. Spectrally Optimal Sampling for Distribution Ray Tracing. Computer Graphics (SIGGRAPH) 25, 4 (1991), 157–164.

[22] Yellott, John I. Jr (1983). Spectral Consequences of Photoreceptor Sampling in the Rhesus Retina. Science. 221 (4608): 382–85. PMID 6867716.

[23] Z. Chen, Z. Yuan, Y. Choi, L. Liu and W. Wang, "Variational Blue Noise Sampling," in IEEE Transactions on Visualization and Computer Graphics, vol. 18, no. 10, pp. 1784-1796, Oct. 2012, doi: 10.1109/TVCG.2012.94.

[24] Morokoff, William J., and Russel E. Caflisch. "Quasi-monte carlo integration." Journal of computational physics 122.2 (1995): 218-230.

[25] Leobacher, Gunther, and Friedrich Pillichshammer. Introduction to quasi-Monte Carlo integration and applications. Switzerland: Springer International Publishing, 2014.

[26] Dick, Josef, and Friedrich Pillichshammer. Digital nets and sequences: discrepancy theory and quasi–Monte Carlo integration. Cambridge University Press, 2010.

[27] Oh, Man-Suk, and James O. Berger. "Adaptive importance sampling in Monte Carlo integration." Journal of Statistical Computation and Simulation 41.3-4 (1992): 143-168.

[28] Press William H., and Glennys R. Farrar. "Recursive stratified sampling for multidimensional Monte Carlo integration." Computers in Physics 4.2 (1990): 190-195.

[29] De Freitas, Nando, and Neil James Gordon. Sequential Monte Carlo methods in practice. Ed. Arnaud Doucet. Vol. 1. No. 2. New York: Springer, 2001.

[30] Del Moral, Pierre. "Mean field simulation for Monte Carlo integration." Monographs on Statistics and Applied Probability 126 (2013): 26.

[31] Uni(corn|form) tool kit, https://utk-team.github.io/utk/. Last access 15th Feb 2022.