EUROGRAPHICS 2021 / N. Mitra and I. Viola
(Guest Editors)

Volume 40 (2021), Number 2

Supplemental: Neural Acceleration of Scattering-Aware Color 3D Printing

Tobias Rittig[†,1], Denis Sumin[†,2], Vahid Babaei[2], Piotr Didyk[3], Alexey Voloboy[4], Alexander Wilkie[1], Bernd Bickel[5], Karol Myszkowski[2], Tim Weyrich[6], Jaroslav Křivánek[1]

[1]Charles University, Czech Republic [2]Max Planck Institute for Informatics, Germany [3]Università della Svizzera italiana, Switzerland
[4]Keldysh Institute of Applied Mathematics RAS, Russia [5]IST Austria, Austria [6]University College London, United Kingdom

1. Introduction

This supplemental material contains multiple topics we could not cover in-depth in the main paper due to space constraints. First, Section 2 describes our dataset and its properties in more details. Details on the network architecture are presented in Section 3. We publish, both, the dataset and the implementation code [RSB*21] alongside the paper. Next, we attempt a speed comparison of our method against a more modern rendering system in Section 4. And finally Section 5 and the following present a study on the behavior of differentiable rendering [NDSRJ20] for reconstructing the albedo of homogeneous scattering media.

2. Training Data

Overall our training set contains 346 volumes with spatially varying scattering and absorption coefficient. These values belong to 5 discrete materials (CMYKW) and have three color channels (RGB) each. Technically this is implemented by storing one index per voxel, and looking up the exact values from a material table. This way, our generated dataset can also be used with a set of different printing materials in the future.

Figure 1 visualizes the collection of shapes our objects represent. Note how sparsely we sample curvature and object rotation. Our weight-sharing approach on coordinate-grid octants relaxes the under-sampling of rotations and thus leaves room in the dataset for other dimensions.

As we state in the paper, the whole dataset does not fit into memory ($>$ 256 GB) during training and reloading large volume chunks considerably hinders training speed. Thus it becomes more important how the dataset content is chosen, as the budget is not infinite. Table 1 lists the exact composition of our dataset. The distribution of values is either coming from our custom volumetric generator combined with halftoning or a plain homogeneous volume in case of the primaries. Six volumes are hand-picked as the validation set.

3. Network Architecture

When talking about network architectures we distinguish between three different variants:

1. $\mathcal{A}_{RPNN}$, the original method by [KMM*17].
2. $\mathcal{A}_{Baseline}$, our baseline including minor adaptations.
3. $\mathcal{A}_{Ours}$, our proposed architecture.

In Table 2 we show their respective properties. In the following we describe technical details on each variant while refering to nomenclature established in Figure 5b and 5c of the main paper. Generally all dense layers have **biases enabled** and **activation disabled** unless otherwise mentioned. Throughout the network a ReLu activation function is used and explicitly marked in the network diagram.

Original Architecture The Multilayer Perceptron (MLP) variant proposed by the authors [KMM*17], consists of the same number of blocks as the stencil hierarchy Σ has levels (K). Each block consists of three dense layers $\mathcal{D}_{[3,5]}$ and processes one level k as input together with information from previous blocks. Although in their paper, they refer to it as two layers, where $\mathcal{D}_3$ and $\mathcal{D}_4$ are combined into one. That is, because $\mathcal{D}_3$ does not have biases. After the blocks, three dense layers $\mathcal{D}_{[6,8]}$ compose the information collected from

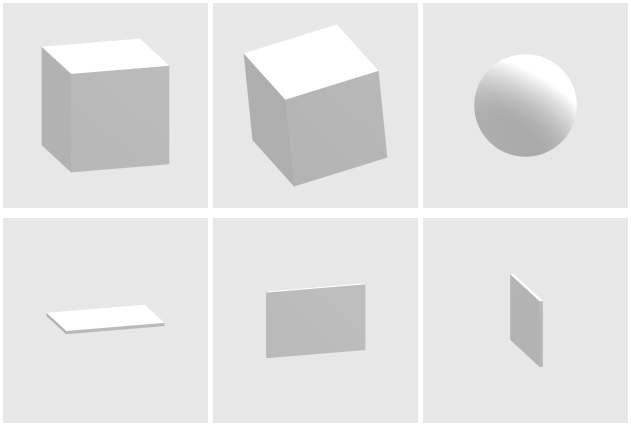


Figure 1: *Collection of shapes we include in our dataset. Each object is included multiple times in different sizes between* 5 mm *to* 20 mm. *Despite curvature being under-represented, the network generalizes even to concave geometry*

Table 1: *Different shapes along with their volumetric material distributions (filling), sizes and numbers used in our dataset of* 346 *objects.*

Shape	Filling	Size / Thickness	Count
Cube	Generator	5 mm	21
		10 mm	21
		15 mm	21
		20 mm	21
	Primaries (CMYKW)	5 mm	5
		10 mm	5
		15 mm	5
		20 mm	5
Sphere	Generator	5 mm	21
		10 mm	21
		15 mm	21
		20 mm	21
	Primaries (CMYKW)	5 mm	5
		10 mm	5
		15 mm	5
		20 mm	5
Cube (rot.)	Generator	5 mm	21
		10 mm	21
		15 mm	21
		20 mm	21
XY plane	Generator	1 mm	6
		2 mm	6
		2.5 mm	6
YZ plane	Generator	1 mm	6
		2 mm	6
		2.5 mm	6
XZ plane	Generator	1 mm	6
		2 mm	6
		2.5 mm	6

the stencils into a single output value. The width of the layers is given by the product of stencil dimensions ($|\Sigma^k| = I$).

The original method separates single-scattering (which is simulated using Monte Carlo (MC)) and only learns multiple-scattering inside the network. The authors also propose to use $K = 10$ levels in the stencil hierarchy for their cloud rendering application.

Baseline Architecture When thinking about a re-implementation within our framework for scattering 3D prints, one quickly notices, that a separate single-scattering simulation is not necessary with such diffuse illumination. Thus, we include this change into our baseline implementation as to avoid any unnecessary computational overhead.

With the considerably higher scattering density of the 3D printing materials when compared to clouds, the highest levels of the stencil have less influence. From experiments, we converged to the parameter $K = 9$ and thus reducing the network capacity.

Table 2: *Feature matrix of architecture variants covered in this paper*

Feature	$\mathcal{A}_{RPNN}$	$\mathcal{A}_{Baseline}$	$\mathcal{A}_{Ours}$
Intended use	clouds	fabrication	fabrication
Lighting	directional	diffuse	diffuse
Per-ray inference	✓	✗	✗
MC single-scattering simulation	✓	✗	✗
Levels K	10	9	9
Spatially-varying parameter	σ_t	σ_s, σ_a	σ_s, σ_a
Trainable layers	33	30	8
Weight sharing over levels ($\equiv_{\mathcal{L}}$)	✗	✗	✓
Weight sharing over octants ($\equiv_{\mathcal{O}}$)	✗	✗	✓

Apart from that, the baseline resembles the original architecture and contains $K * 3 + 3 = 30$ dense layers.

Our Architecture In our architecture we introduce two weight-sharing schemes that are motivated based on volumetric light transport. First of all, all blocks share the weights with each other ($\equiv_{\mathcal{L}}$), and only the inputs (scattering and absorption coefficients) are scaled for physical correctness. Second, we introduce weight-sharing over symmetric parts of the stencil Σ to allow for rotational invariance ($\equiv_{\mathcal{O}}$). The latter is applied twice througout the architecture, once for processing inputs $\mathcal{D}_{[1,2]}$ and for processing the outputs $\mathcal{D}_{[6,8]}$.

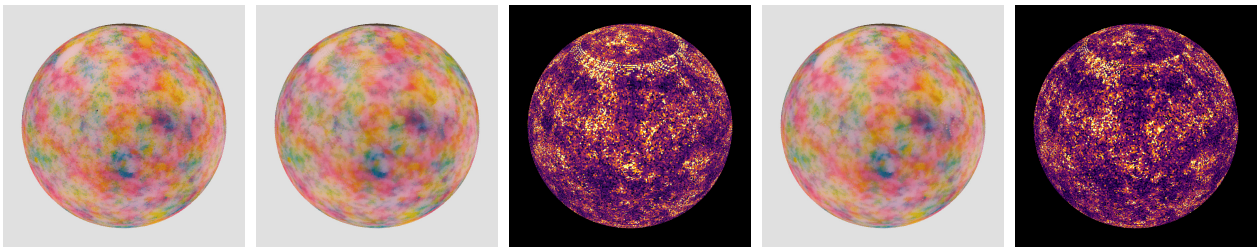
When processing octants, we slice the stencils as described in Figure 6b in the main paper, rotate them to a common alignment and apply the corresponding layers to the eight parts. After concatenation, the network flow continues in dense layers with a wider width, due to the overlaps in the slicing.

In total, our architecture consists of only eight trainable layers.

Figure 2 shows an extended version of Fig 13 of the main paper, where spectral prediction is shown for the baseline architecture.

4. Speed Comparison

For obtaining an impression on the relative speed of our method versus a more modern rendering system, we conducted an experiment and compared against Mitsuba2 [NDVZJ19] (on commit `3214250`). In the main paper, we compare on more results to an improved volume path-tracer based on [Jak10]. Here we take a single object (YELLOW VASE) and manually re-render it using various rendering systems and configurations. Due implementation differences we cannot render a full surface prediction based on a custom camera model, but for speed comparisons we deem it sufficient to render a perspective image with narrow field of view, where every pixel hits the volume. We render a 128×128 image with 512 spp,



(a) *MC 128 spp* **(b)** *Baseline* **(c)** *Difference* **(d)** *RPNN (Ours)* **(e)** *Difference*

Figure 2: *Extended version of Fig. 13 of the main paper for a baseline comparison. Spectral predictions for an object formed by virtual spectrally-defined materials. Our network generalizes well over scattering parameters from only 15 discrete values and can accurately predict continuous spectral curves. CIE dE 2000: 0*  *10*

Table 3: *Timings for a single forward prediction of the* YELLOW VASE *(Iteration 10). The relative speed for [NDVZJ19] is calculated between the mean and maximum (as opposed to the sum) of all three color channels. Despite the difference in hardware setup, the relative speed is comparable to Table 1 in the main paper.*

Method	Intersection	Red	Green	Blue	Relative Speed
[Jak10]	k-d tree	897 s	924 s	1006 s	0.517×
[SRB*19]	k-d tree	483 s	475 s	502 s	1×
[NDVZJ19] `gpu_mono`	OptiX7+RTX	11 354 s	5652 s	14 732 s	0.099–0.138×
[NDVZJ19] `scalar_mono`	k-d tree	2102 s	489 s	526 s	0.695–1.406×
	Embree	1745 s	217 s	241 s	0.837–1.990×
[NDVZJ19] `packet_mono`	k-d tree	2769 s	636 s	762 s	0.528–1.052×
	Embree	2367 s	288 s	342 s	0.617–1.462×
Homogeneous Medium					
[NDVZJ19] `gpu_mono`	OptiX7+RTX	162 s	163 s	162 s	8.984–8.996×
[NDVZJ19] `scalar_mono`	k-d tree	73 s	71 s	70 s	20.037–20.496×
	Embree	62 s	63 s	61 s	23.144–23.512×
[NDVZJ19] `packet_mono`	k-d tree	76 s	77 s	76 s	19.095–19.157×
	Embree	12 s	12 s	12 s	119.127–120.063×

the `volpath` integrator (where applicable) and a maximum path-length of 1000 (Russian Roulette disabled). Allowing for a comparison with Table 1 in the main paper, we calculate relative speeds compared to the same reference implementation.

As the previous version of Mitsuba [Jak10] does not support a spectrally-varying extinction coefficient, a common approach is to run three independent renderings for each color channel. That is the approach of [SRB*19] and also the one our Radiance Predicting Neural Network (RPNN) is based on. Forming matters more complicated, Mitsuba2 [NDVZJ19] does support spectrally-varying extinction coefficients. We still report times for three independent renderings and assume the overall performance to lie between the mean and the maximum. Given the dominant factor is delta tracking for the heterogeneous medium, this is a sensible conclusion.

CPU algorithms are run with 32-threads on two Intel E5-2680 v3 CPUs (AVX2 enabled, 8-wide SIMD) using the built-in k-d tree or Embree for ray intersection. On the GPU, renderings used Nvidia Optix7 on a Nvidia TITAN RTX with 24GB memory of which 15GB were occupied. The experiments were run on a different hardware setup than the results obtained in the main paper. We observe a ∼2.7× slow down when switching environments. Still, the measurements are performed consistently for all algorithms and relative speed factors are comparable to our main results.

The resulting timings are listed in Table 3. We observe that the native Mitsuba `volpath` integrator [Jak10] is about 2× slower due to inefficient emitter sampling through a dielectric interface.

A prominent observation is that GPU rendering is considerably slower than CPU rendering in Mitsuba2. That is an observation that others also made previously [Wa20] and is attributed by the authors to the wavefront approach. It also seems most affected by different densities of the three channels.

Rendering on the CPU is likely to be on par with [Jak10].

When switching the medium to be a homogeneous scatterer with textured albedo, the rendering times improve drastically. This measurement is important to keep in mind for the upcoming sections.

5. Differentiable Rendering Optimization Setup

After the publication of several differentiable volume rendering papers [NDVZJ19, ZWZ*19, NDSRJ20] in recent years, the land-

scape of volumetric reconstruction techniques changed considerably. With the most recent work [NDSRJ20], an application becomes finally computationally tractable. In the following, we compare this line of research to the state-of-the-art heuristic method in 3D printing [SRB*19].

The experimental setup of this comparison differs from the rendering setup usually employed in scattering compensation [ESZ*17, SRB*19]. We inherit this setup from the published code of [NDSRJ20] which is based on an older version of Mitsuba2. After consultation with the authors, we were advised to replicate their setup in our pipeline for the sake of this comparison. We consider the construction of a genuine 3D printing optimization setup using Radiative Backpropagation (RB) outside the scope of this paper and leave it for future work.

Table 4: *Properties of the rendering setup used for RB and the heuristic refinement whenever it is compared to RB. The original setup employed in previous 3D printing work [SRB*19] is shown (but not used).*

Property	[NDSRJ20]	[SRB*19]
Illumination	diffuse white	diffuse white
Media Modeling	homogeneous	heterogeneous
Albedo	varying	varying (discrete)
Density	constant	varying (discrete)
Phase function	isotropic	HG ($g = 0.4$)
Halftoning	no	yes
Camera	perspective	surface voxels aligned with surface normal

We list the differences between the two approaches in Table 4. The most salient differences are the modeling, the phase function and the lack of material discretization.

Discretization Halftoning discretizes printer materials to ensure the ability to fabricate the object's appearance. The method of [SRB*19] includes halftoning in every iteration, to predict the appearance of the actual printer material arrangement. Including a discretization step would also be possible in a differentiable framework, however it would introduce another source of discontinuity that might need special treatment. Leaving physical printability aside, one can still compare the methods for virtual objects only. Therefore, a continuous volume albedo is being reconstructed to match a target surface albedo.

Density For the density of the medium, one can assume a constant that lies somewhere in the order of the discrete materials. Upon review of the published code, we found that figures in [NDSRJ20] were produced with a considerably high density of $75\,\mathrm{mm}^{-1}$. That helps obtaining a near perfect match towards the target images and the lack of visible lateral scattering. In the comparisons here, we test a range of different, lower densities ranging from $1.125\,\mathrm{mm}^{-1}$ to $9\,\mathrm{mm}^{-1}$.

Phase function The difference in phase function is particularly severe for the appearance of translucent media [GXZ*13]. Images with an isotropic phase function exhibit less lateral scattering than a forward scattering phase function. Inspired by similarity theory, we reduced the density for some experiments to regain some level of unwanted lateral scattering. This is important for the comparison, as it is a very important reason for the existence of scattering compensation in 3D printing.

Modeling The way a medium is modeled in a renderer influences mostly the efficiency of the estimator. In the setup of [NDSRJ20], the medium is modeled as a homogeneous medium with constant density and spatially-varying albedo. This allows for closed-form free-flight sampling while still having a textured appearance. Limited by the heuristic pipeline implementation [SRB*19], which is based on [Jak10], we can only model the medium as fully heterogeneous but with constant density values.

The main implication of that is a limited comparability of timings between the two implementations. Besides the advantage of Mitsuba2 being a more modern renderer executed on GPU, it thus also benefits from more efficient volume sampling. We refrained from artificially slowing down Mitsuba2 by using a heterogeneous medium to keep a high experimentation speed. From experiments in Table 3 we can conclude the difference in speed. In the main paper, we thus only refer to iteration counts and omit the comparison of wall-clock timings of individual iterations. Theoretically both methods could be driven by the same MC implementation alleviating the necessity for absolute time measurements.

Scene Geometry The slab is modeled to consist of $256 \times 256 \times 64$ voxels at 300 dpi resolution. This makes it $\approx 21.6\,\mathrm{mm} \times 21.6\,\mathrm{mm} \times 5.4\,\mathrm{mm}$ in size. The illumination is a constant white environment.

Optimization Procedure The differentiable rendering setup uses the Adam optimization algorithm [KB14] with default settings to update the parameters. The sample count is kept constant throughout the optimization for both forward and backward pass to 128 spp except for the last iteration which is set to 512 spp. We are aware of obvious speed-up techniques such as lowered sample count, dynamic sample count and learning rate adjustments but chose not to apply them here for a fair comparison on best possible quality. As we argue in Section 8, these techniques alone would not have influenced the outcome of the speed considerations of this comparison.

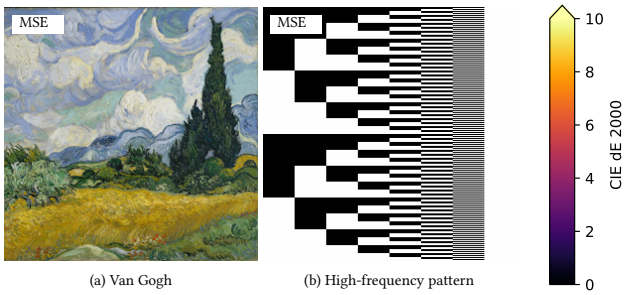
The heuristic setup predicts with 128 spp or 512 spp depending on the experiment whereas the last iteration is always 512 spp.

6. Initializations

With gradient descent being a local optimization method, the initial condition influences the trajectory of the optimization. We tested various configurations to initialize the voxels including constant colors, noise, extrusions, and the heuristic solution. Figure 3 depicts the two target images.

6.1. Van Gogh

Each row in Figure 9 lists the tested initializations whereas columns depict different iterations throughout the optimization from 1 to 300. A cutout on the trees highlights the contrast towards the sky



(a) Van Gogh (b) High-frequency pattern

Figure 3: *Targets (a) and (b) used in the study. Insets on the top left denote MSE values. The CIE dE 2000 difference images are colored using the inferno colormap (lower/darker is better).*

and manifestation of details within the trees' texture. The difference images underline the spatial distribution of error over the image.

Convergence Overall we observe a diversity of convergence speed between the initializations. That is visible from the graphs in Figure 4 and visually in the images of individual iterations. BLACK is expectably bad for initialization as no albedo gradients will have a greater magnitude than zero. WHITE is the slowest of all. A medium GREY seems to be better for convergence and final result quality. NOISE starts only slightly slower, but converges to equal high quality. Extrusions of 1 LAYER and 2 LAYER are good initializations for convergence but lack slight crispness after 300 iterations. Finally, a pre-converged solution (HEURISTIC) gives the lowest overall starting error and only changes slightly in MSE while staying perceptually constant. We attribute this slight change in MSE partially to the mismatch in camera setup but mostly to the different error metric.

Final quality As for final image quality after 300 iterations, GREY, NOISE, and HEURISTIC deliver the most details. 2 LAYER shows crisper details than 1 LAYER and WHITE looks overall washed out. The perceptual quality differences are not reflected in the MSE error values in the insets. We suspect the quality might reach the

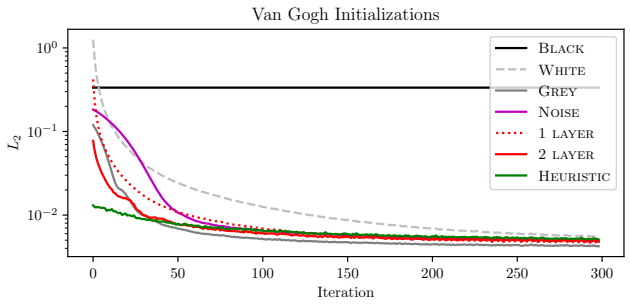


Figure 4: *Convergence plot of different initializations for Radiative Backpropagation (RB) [NDSRJ20] on the Van Gogh target. The* HEURISTIC *initialization is the result of a 512 spp heuristic refinement [SRB*19] presented in Figure 11.*

same level with all initializations, but would require more iterations depending on convergence speed.

Volumetric Arrangement The final volumetric arrangement of the albedo values in the last column shows the most saliency between the initialization methods. Throughout, the topmost layer looks like an unsharp-masked version of the target image. Layers below are less sharp but exhibit a form of morphological dilation of bright (white) regions. In even deeper layers this widening of white tapers off into the untouched initial values. This visualizes the shape of a subsurface scattering kernel (with orthogonal incidence) that one can imagine in form of a carrot.

The high albedo of WHITE allows paths with high contribution to reach deep into the medium. Thus, the optimizer initially distributes improvement energy over more voxels into deep layers. When more absorption is accumulated on the surface, these deeper layers loose influence. This is mirrored in the slower convergence speed of the WHITE initialization.

GREY initially has a constant absorption everywhere, which results in a more compressed absorption towards the surface. This is visible from the maximum reach of white material in depth as well as from the start of widening from the top. Maximum absorption is not accumulated in the top-most layer but spread over the top two to three.

NOISE is even more compressed than GREY but still does not accumulate the peak absorption in a single layer. In 1 LAYER the topmost layer is very absorbing and focused while the other layers behave like WHITE. 2 LAYER shows identical tendencies but with the absorption being less compressed and spread over two layers.

As the heuristic approach [ESZ*17,SRB*19] is designed to maximize absorption close to the surface it generally does not reach very deep. Here, only five to six layers are covered considerably and deeper layers exhibit more sparse noise. RB performs only minor updates to the volume over the course of 300 iterations.

6.2. High-frequency Pattern

As the above color optimization is essentially three independent per-channel optimizations we isolate contrast perception from color perception in a black and white target. We constructed the target visible in Figure 3 (b) to contain multiple spatial frequencies and feature larger constant areas, as well as areas with fine details. Again, Figure 10 shows different initializations per row and the convergence graphs for this target are visible in Figure 5. We observe overall very similar behavior as in Section 6.1 and will describe only special considerations here.

On first glance, the EXTRUSION initialization already produces a perfectly acceptable result on the first iteration. The optimizer however tries to reduce the lateral scattering into the bright white values first. On these kind of hard edges, it is a tradeoff between light scattering in dark or the other way round. Here, the optimizer is very on-sided in favor of light values leaving strong lateral scattering into the black parts. This can be traced back to the used metric.

In terms of metric values, the convergence graph also shows a preference for bright initializations (WHITE, 1 LAYER, 2 LAYER).

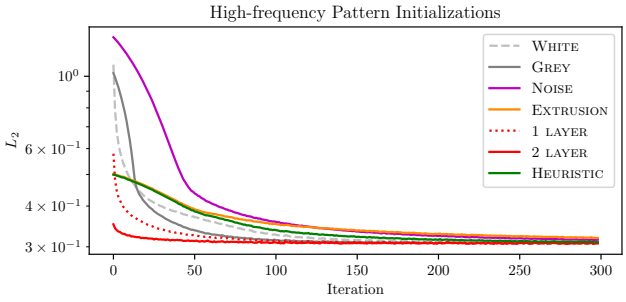


Figure 5: *Convergence plot of different initializations for Radiative Backpropagation (RB) [NDSRJ20] on high-frequency target. The* HEURISTIC *initialization is the result of a 512 spp heuristic refinement [SRB*19] presented in Figure 12.*

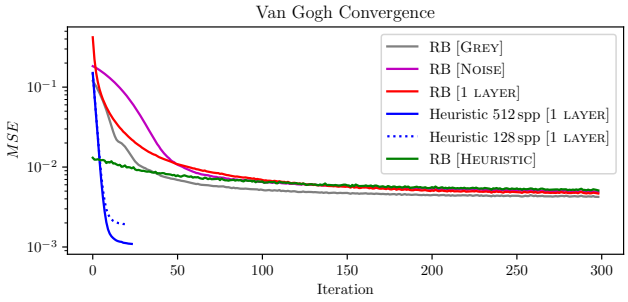


Figure 6: *Convergence plot for comparison between Radiative Backpropagation (RB) [NDSRJ20] and heuristic refinement [SRB*19] presented in Figure 11 on Van Gogh target image.The heuristic solution requires one order of magnitude less iterations than stochastic gradient descent (SGD).*

With the metric not focusing on dark areas it is also harder to reproduce a true black color. In this comparison it is only achieved, where the EXTRUSION and HEURISTIC initializations already had perfect black albedo.

All slices in Figure 10 show aliasing artifacts on higher frequencies from misalignment of camera pixels and volume voxels due to the perspective camera.

7. Heuristic Comparison

Explicit comparison between the two methods is visualized in Figs. 11 and 12. We take the most interesting initializations for Radiative Backpropagation (RB) and show them alongside two optimization runs of the heuristic approach. The RB results are thus not explicitly described again and we focus instead on the differences to the heuristic run.

7.1. Van Gogh

Judging from the difference images, the GREY and NOISE runs achieve an equal fidelity of tree details as the heuristic approach with 512 spp. Edge sharpness is slightly better for the heuristic approach, which also shows in the hard edges of the clouds.

The heuristic approach run on a lower sample-count does not manifest visible artifacts into the medium that would show up when re-rendered with equal spp.

When comparing the two methods on an equal iteration number one can easily spot the difference in convergence behavior. After only 23 iterations, RB exhibits a very schematic appearance. Considerably more iterations even in flat regions of the convergence graph (Figure 6) are required to develop fine texture details.

7.2. High-frequency Pattern

For the high-frequency target the comparison is shown in Figure 12. Here, the heuristic solution produces a much sharper solution than RB. Contrast is preserved even for very high frequencies. For that, it occupies more layers than any solution of RB and propagates absorption as deep as necessary to achieve a perfect black on the surface. Continuing the heuristic solution (e) with RB (g) introduces more white scattering into the black parts and decreases edge sharpness.

In terms of convergence, the plot in Figure 7 again shows less required iterations for the heuristic. This plot also visualizes how perceptual quality and error metric values are clearly not aligned. The minimum lies in very early iterations where most of the image is bright and the pattern is hardly perceptible. After that, lateral scattering increases the error, but the contrast of the pattern increases which is perceptually preferable.

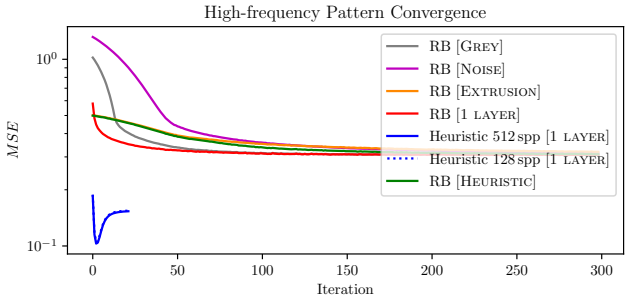


Figure 7: *Convergence plot for comparison between Radiative Backpropagation (RB) [NDSRJ20] and heuristic refinement [SRB*19] presented in Figure 12 on high-frequency target.The heuristic solution requires one order of magnitude less iterations than SGD. The increase in error is from black material scattering into white.*

8. Discussion

Our results show, that the performance gap with the more specialized, heuristic approach is one order of magnitude. As we note in Section 5, differentiable rendering can benefit from very low sample-counts to improve the absolute speed. A fine-tuning of the SGD parameters might also yield a performance boost as well. Overall, further work is necessary to investigate the convergence behavior of differentiable rendering.

Metric This comparison inherited the simple L_2 norm optimization from the original code publication [NDSRJ20]. As it is obvious from numerous results (eg. Figure 9 (d) vs (e)), a lower MSE does not necessarily correlate with perceptual preferences. We identify the choice of metric to be important to reach higher quality. Especially for 3D surfaces, a new metric with lateral support is necessary. The heuristic approach has some perceptual aspects build-in as it tries to maximize local contrast.

Differentiable rendering can be more general in terms of illumination. Where the heuristic is designed for diffuse illumination, the more general tool can incorporate arbitrary scene complexity if a specific application requires that.

References

[ESZ*17] ELEK O., SUMIN D., ZHANG R., WEYRICH T., MYSZKOWSKI K., BICKEL B., WILKIE A., KŘIVÁNEK J.: Scattering-aware Texture Reproduction for 3D Printing. *ACM Transactions on Graphics (Proc. SIGGRAPH Asia) 36*, 6 (Nov. 2017), 241:1–241:15. doi:10.1145/3130800.3130890. 4, 5

[GXZ*13] GKIOULEKAS I., XIAO B., ZHAO S., ADELSON E. H., ZICKLER T., BALA K.: Understanding the role of phase function in translucent appearance. *ACM Transactions on Graphics 32*, 5 (Sept. 2013), 147:1–147:19. doi:10.1145/2516971.2516972. 4

[Jak10] JAKOB W.: Mitsuba renderer, 2010. http://www.mitsuba-renderer.org. 2, 3, 4

[KB14] KINGMA D. P., BA J.: Adam: A method for stochastic optimization. *arXiv:1412.6980 [cs]* (Dec 2014), 1–11. arXiv: 1412.6980. URL: http://arxiv.org/abs/1412.6980. 4

[KMM*17] KALLWEIT S., MÜLLER T., MCWILLIAMS B., GROSS M., NOVÁK J.: Deep Scattering: Rendering Atmospheric Clouds with Radiance-Predicting Neural Networks. *ACM Transactions on Graphics (Proc. SIGGRAPH Asia) 36*, 6 (Nov. 2017), 231:1–231:11. doi:10.1145/3130800.3130880. 1

[NDSRJ20] NIMIER-DAVID M., SPEIERER S., RUIZ B., JAKOB W.: Radiative Backpropagation: An Adjoint Method for Lightning-Fast Differentiable Rendering. *ACM Transactions on Graphics (Proc. SIGGRAPH) 39*, 4 (July 2020), 146:1–146:15. doi:10.1145/3386569.3392406. 1, 3, 4, 5, 6, 7, 9, 10, 11, 12

[NDVZJ19] NIMIER-DAVID M., VICINI D., ZELTNER T., JAKOB W.: Mitsuba 2: A Retargetable Forward and Inverse Renderer. *ACM Transactions on Graphics 38*, 6 (Nov. 2019), 203:1–203:17. doi:10.1145/3355089.3356498. 2, 3

[RSB*21] RITTIG T., SUMIN D., BABAEI V., DIDYK P., VOLOBOY A., WILKIE A., BICKEL B., MYSZKOWSKI K., WEYRICH T., KŘIVÁNEK J.: Neural Scattering Prediction: Code and Dataset, 2021. URL: https://github.com/denis-sumin/neural-scattering-prediction. 1

[SRB*19] SUMIN D., RITTIG T., BABAEI V., NINDEL T., WILKIE A., DIDYK P., BICKEL B., KŘIVÁNEK J., MYSZKOWSKI K., WEYRICH T.: Geometry-Aware Scattering Compensation for 3D Printing. *ACM Transactions on Graphics (Proc. SIGGRAPH) 38*, 4 (July 2019), 111:1–111:14. doi:10.1145/3306346.3322992. 3, 4, 5, 6, 11, 12

[Wa20] WALKOM G., ABHINAVVS: Mitsuba2 github issues, Apr. 2020. Accessed 2020-10-04. URL: https://github.com/mitsuba-renderer/mitsuba2/issues/72. 3

[ZWZ*19] ZHANG C., WU L., ZHENG C., GKIOULEKAS I., RAMAMOORTHI R., ZHAO S.: A differential theory of radiative transfer. *ACM Transactions on Graphics (Proc. SIGGRAPH Asia) 38*, 6 (Nov. 2019), 227:1–227:16. doi:10.1145/3355089.3356522. 3

Figure 8: *Comparison of prediction methods inside a refinement loop: (a) target models (b) using baseline RPNN-based predictions, and (c) our proposed solution using RPNN-based predictions.*

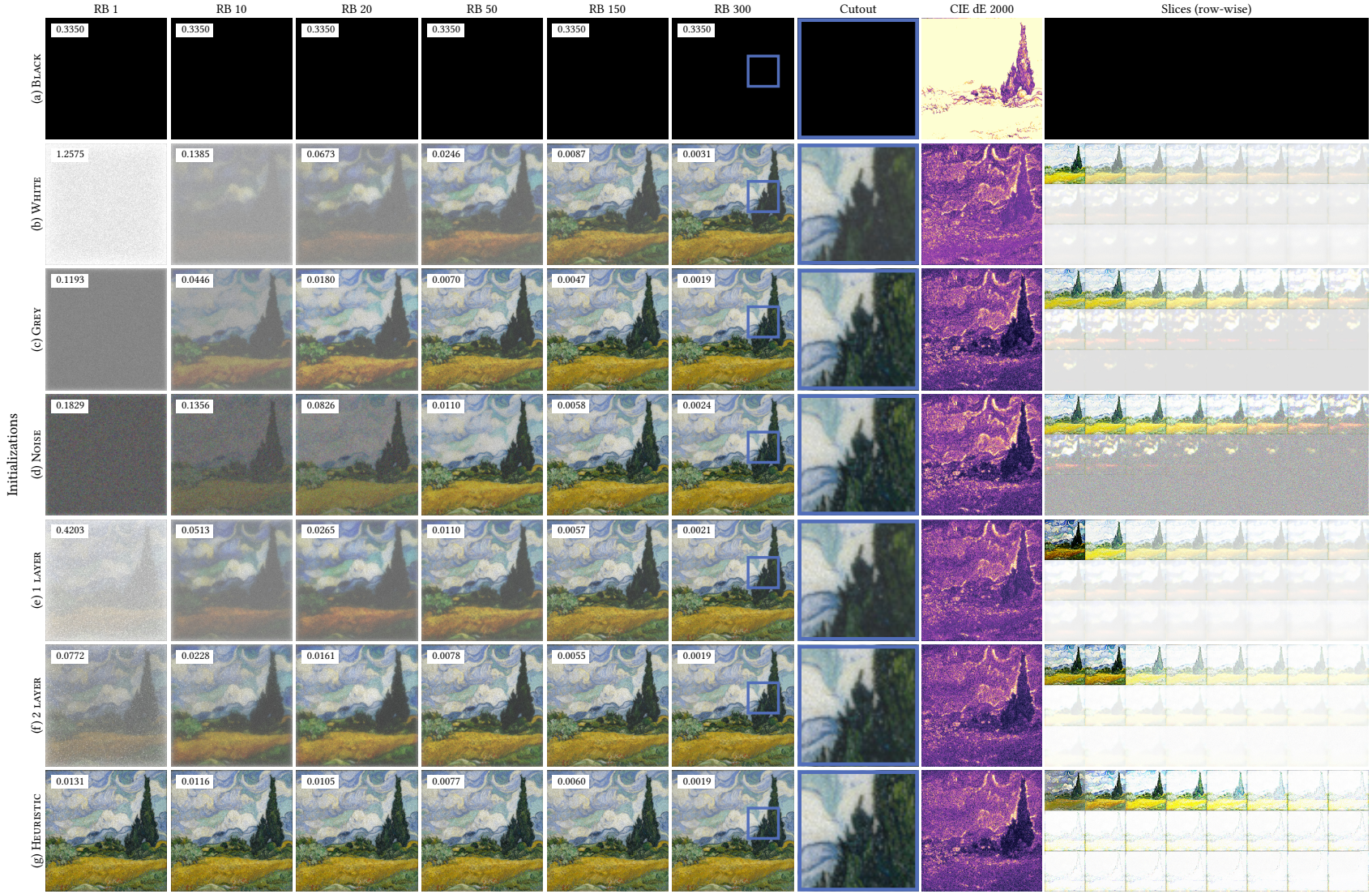
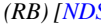


Figure 9: *Comparing Radiative Backpropagation (RB) [NDSRJ20] over different volume initializations (a-g) on density* $4.5\,\mathrm{mm}^{-1}$ *and Van Gogh target image. Here,* HEURISTIC*is used as an initialization and further optimized with RB. Formalities are explained in Figure 3. Beware, that error values and images for the last iterations are computed with* $4\times$ *higher (512) spp for comparability. CIE dE 2000: 0*  *10*

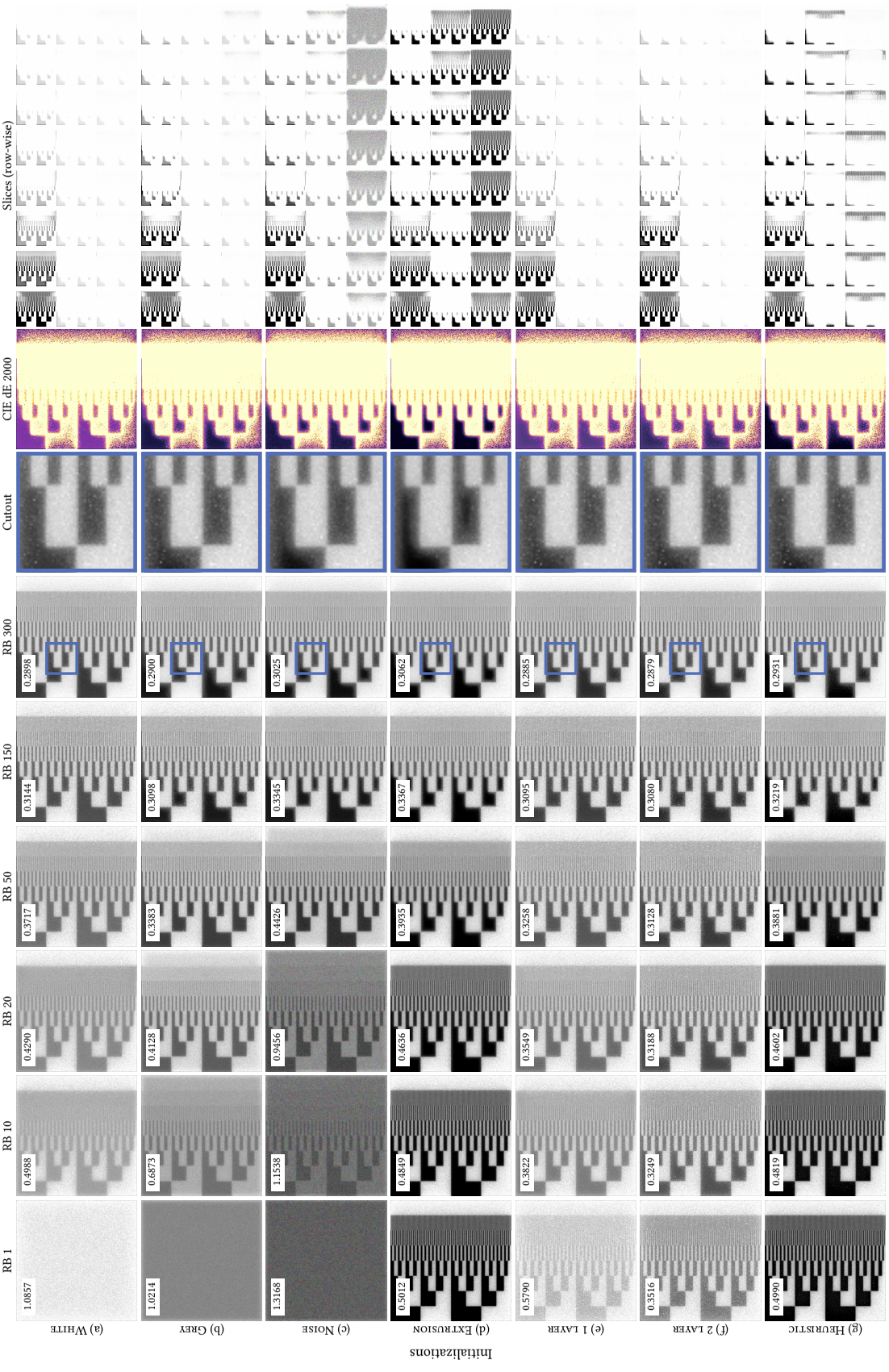


Figure 10: *Comparing Radiative Backpropagation (RB) [NDSRJ20] over different volume initializations (a-g) on density* $4.5\,\mathrm{mm}^{-1}$ *and a high-frequency pattern. Here,* HEURISTIC*is used as an initialization and further optimized with RB. Formalities are explained in Figure 3. Beware, that error values and images for the last iterations are computed with* $4\times$ *higher (512) spp for comparability. CIE dE 2000:* 0 10

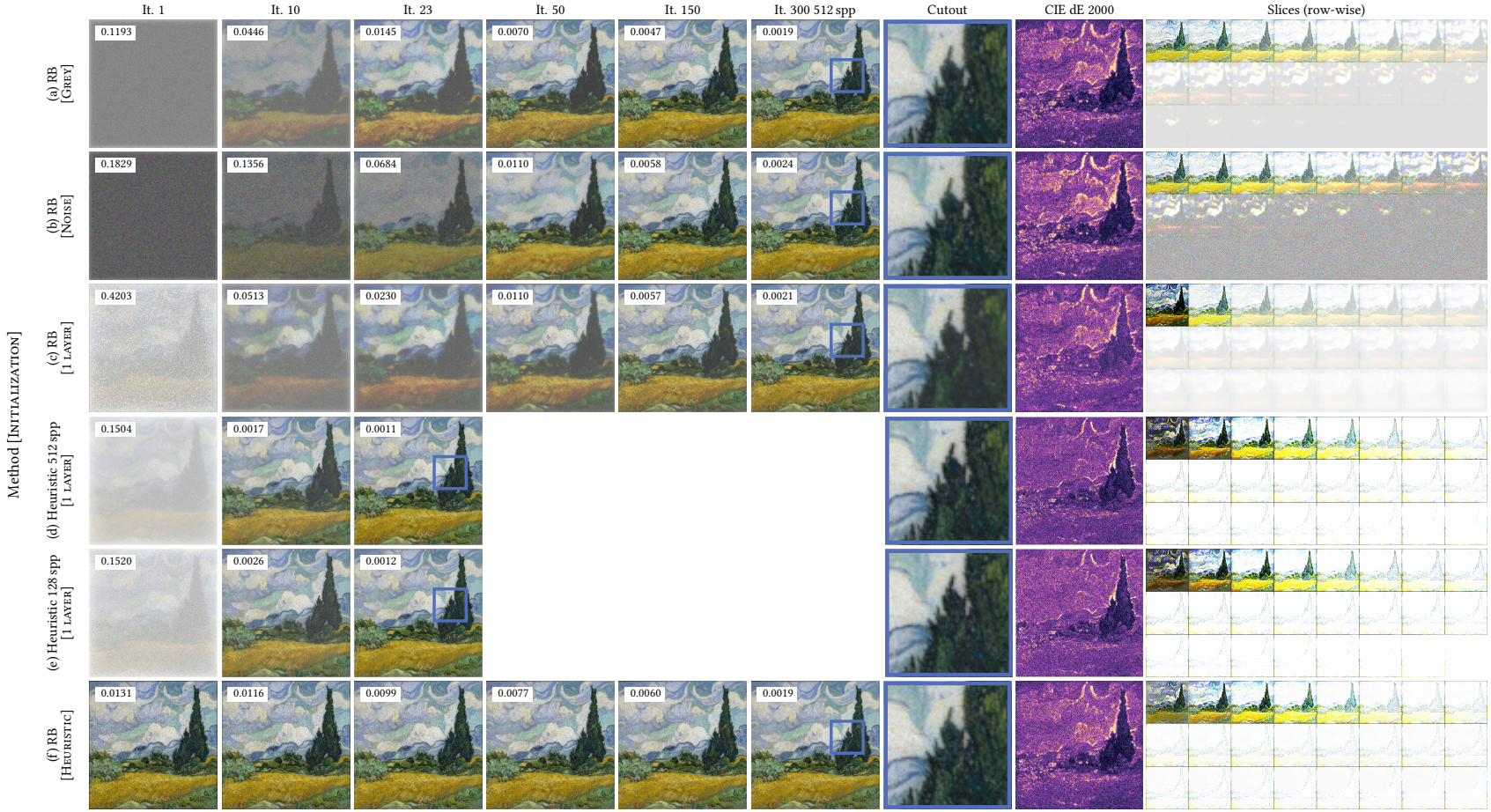
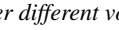


Figure 11: *Comparing Radiative Backpropagation (RB) [NDSRJ20] (a-c,f) and a heuristic approach [SRB*19] (d,e) over different volume initializations and sample count for density* $4.5\,\mathrm{mm}^{-1}$ *and Van Gogh target image. In the last row,* HEURISTIC*is used as an initialization and further optimized with RB. Formalities are explained in Figure 3. Beware, that error values and images for the last iterations are computed with* $4\times$ *higher (512) spp for comparability. CIE dE 2000: 0*  *10*

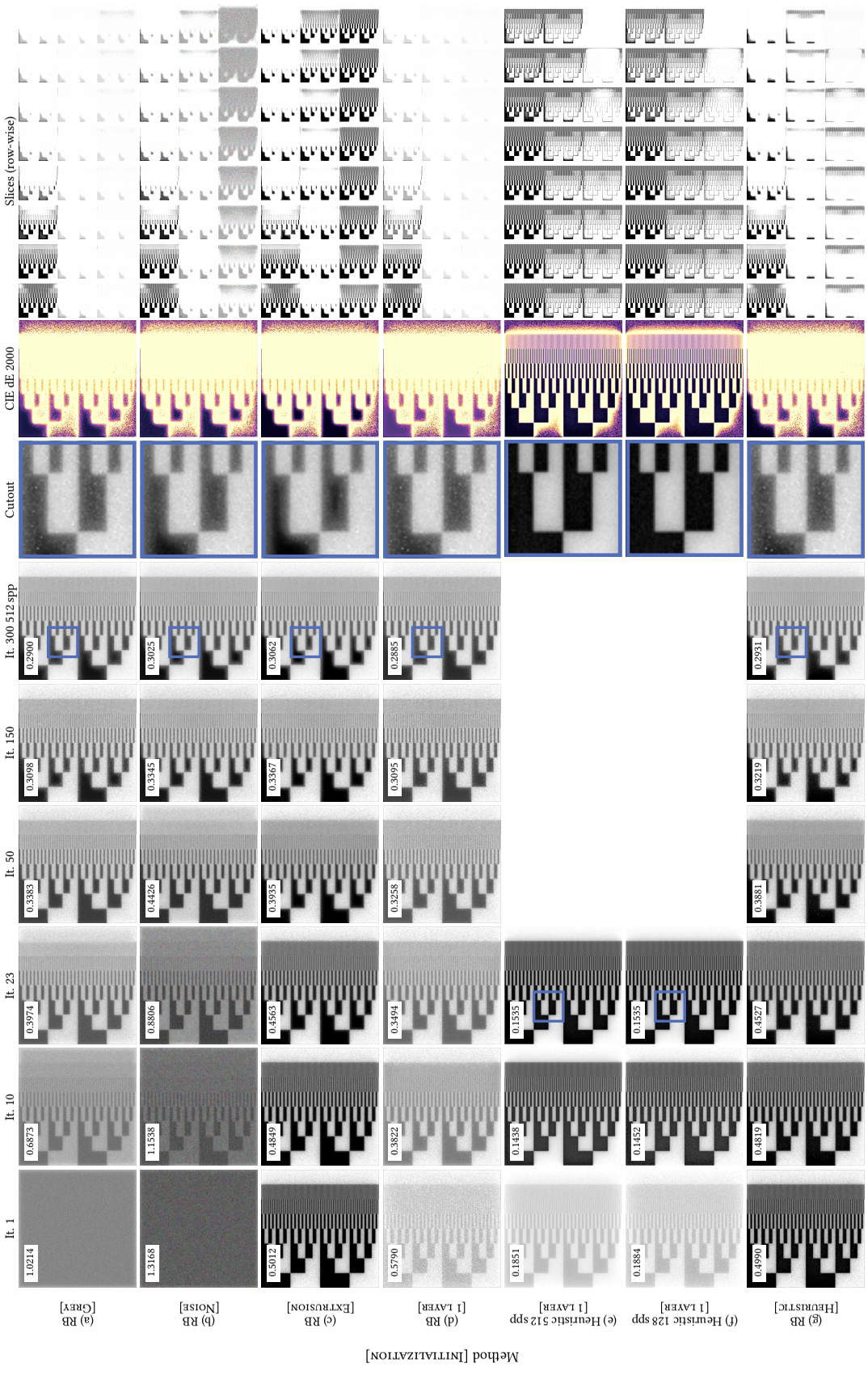


Figure 12: *Comparing Radiative Backpropagation (RB) [NDSRJ20] (a-d,g) and a heuristic approach [SRB*19] (e,f) over different volume initializations and sample count for density* $4.5\,\mathrm{mm}^{-1}$ *and a high-frequency pattern. In the last row,* HEURISTIC*is used as an initialization and further optimized with RB. Formalities are explained in Figure 3. Beware, that error values and images for the last iterations are computed with* $4\times$ *higher (512) spp for comparability. CIE dE 2000: 0* *10*