Multi-Dimensional Procedural Wave Noise - Supplemental Material

PASCAL GUEHL, LIX, Ecole Polytechnique, CNRS, Institut Polytechnique de Paris, France
RÉMI ALLÈGRE, ICube, Université de Strasbourg, CNRS, France
GUILLAUME GILET, Université de Sherbrooke, Canada
BASILE SAUVAGE, ICube, Université de Strasbourg, CNRS, France
MARIE-PAULE CANI, LIX, Ecole Polytechnique, CNRS, Institut Polytechnique de Paris, France
JEAN-MICHEL DISCHLER, ICube, Université de Strasbourg, CNRS, France

This supplemental material contains a more exhaustive theoretical and practical comparative study between Gabor sparse convolution noise and wave noise in the 3D case. We also show additional results in 2D, 3D and 3D+t.

CCS Concepts: • **Computing methodologies** → **Texturing**.

Additional Key Words and Phrases: Procedural noise

ACM Reference Format:
Pascal Guehl, Rémi Allègre, Guillaume Gilet, Basile Sauvage, Marie-Paule Cani, and Jean-Michel Dischler. 2025. Multi-Dimensional Procedural Wave Noise - Supplemental Material. *ACM Trans. Graph.* 44, 4 (August 2025), 18 pages. https://doi.org/10.1145/3730928

1 Extended Comparative Study

1.0.1 Theoretical comparison. We formally review how current procedural approaches generate noise, and how they comply with the properties (a) to (f) described in the paper section 3. Lattice, sparse convolution, spectral (LRP) and tile-based (RTB) noises (see survey section 2), share three common characteristics:

- An infinite set of discrete points $\mathbf{p}_k$ derived from a periodic tesselation of $\mathbb{R}^n$ using space-filling polytopes. The $\mathbf{p}_k$ are either the polytope vertices (as for lattice [Perlin 1985, 2001], LRP [Gilet et al. 2014], RTB [Heitz and Neyret 2018] noises) or points randomly generated inside the polytopes (as for sparse convolution noises [Lewis 1984, 1989] [Van Wijk 1991] [Galerne et al. 2012, 2017; Lagae et al. 2009] [Cavalier et al. 2019]).
- A spatial kernel $K(\mathbf{x}, \mathbf{p}_k)$ that can be an analytic function (a Gaussian [Lewis 1989], Gabor [Lagae et al. 2009], sinc function [Gilet et al. 2012]), or an image [Galerne et al. 2017; Heitz and Neyret 2018], or a constant random value [Perlin 1985]. Usually, K applies in one way or another a pseudo-random number generator (PRNG), with its seed set according to the $\mathbf{p}_k$ to provide randomness, yet ensuring determinism.
- A weighted sum. The weights may be constant or involve an interpolation function w further depending on offsets $\mathbf{x} - \mathbf{p}_k$, as for RTB [Heitz and Neyret 2018] and lattice noises [Perlin 1985, 2001].

We unify them under the term *point-based noises* $\mathcal{N}_p$, and define them with a single common formula:

$$\mathcal{N}_p(\mathbf{x}) = \sum_{k=1}^{n_k} w_k(\mathbf{x}) K_k(\mathbf{x}) \tag{1}$$

Properties of $\mathcal{N}_p$: Equation 1 is *continuous (Prop.a)* if $w_k K_k$ is continuous, a property of analytic functions but not of images, unless reconstructed with filters, which might however introduce artifacts. Sparse convolution can produce discontinuities when infinite kernels are clamped, like Gaussians. These are typically imperceptible, provided the variance of the Gaussians is chosen correctly w.r.t. the size of the underlying polytopes. Therefore grid size and Gaussian variance are closely linked with Lewis, Gabor and Phasor noises. *Computational complexity (Prop.b)* is proportional to n_k and remains constant if n_k is bounded and $w_k K_k$ operates in $\Theta(1)$. Polytopes are specifically used to constrain n_k. *Memory cost (Prop.c)* depends only on K_k, as polytopes and $\mathbf{p}_k$ are never explicitly stored in a procedural setup. *Spatial unboundedness (Prop.e)* arises from periodic polytopes, while pseudo-random number generation (PRNG) ensures *non-periodicity (Prop.d)*. It is worth noting that the inherent periodicity of PRNGs is typically not a concern, as their period is vastly larger than the domain over which the noise is usually applied. When K_k depicts visually salient characteristics and n_k is small, repetition artifacts may occur, which happens for example when RTB [Heitz and Neyret 2018] is diverted to generate structured textures instead of noise. *Higher dimensions (Prop.e)* are obtained by extending the dimension of the polytopes and the K_k, which is straightforward with functions, like Gaussians. *Random access (Prop.f)* is supported, since all $\mathbf{p}_k$ and K_k are completely independent.

Concerning *spectral control (Prop.g)*: Equation 1 defines noise by convolution with a point distribution, reducing to a sum due to sparsity. The Fourier Transform (FT) of $\mathcal{N}_p$ is therefore a product between $\mathcal{F}(w_k K_k)$ and $\mathcal{F}(\mathbf{p}_k)$, with the product $w_k K_k$ becoming a convolution in spectral domain. For lattice noises [Perlin 1985], K_k is a constant, so the PSD only depends on w_k and $\mathbf{p}_k$, making spectral control difficult. For other noises, spectral control is mainly linked to $\mathcal{F}(K_k)$, though the PSD also depends on $\mathbf{p}_k$ and w_k. Because randomly weighted Poisson point processes correspond to a

Authors' Contact Information: Pascal Guehl, LIX, Ecole Polytechnique, CNRS, Institut Polytechnique de Paris, France, pascal.guehl@polytechnique.edu; Rémi Allègre, ICube, Université de Strasbourg, CNRS, France, remi.allegre@unistra.fr; Guillaume Gilet, Université de Sherbrooke, Canada, Guillaume.Gilet@USherbrooke.ca; Basile Sauvage, ICube, Université de Strasbourg, CNRS, France, sauvage@unistra.fr; Marie-Paule Cani, LIX, Ecole Polytechnique, CNRS, Institut Polytechnique de Paris, France, marie-paule.cani@polytechnique.edu; Jean-Michel Dischler, ICube, Université de Strasbourg, CNRS, France, dischler@unistra.fr.

ACM 1557-7368/2025/8-ART
https://doi.org/10.1145/3730928

white noise approximation, whose theoretical FT is constant, sparse convolution noise using a spectrally controllable kernel (Gabor [Lagae et al. 2009], Texton [Galerne et al. 2017]), achieve strong spectral control. RTB [Heitz and Neyret 2018] and LRP [Gilet et al. 2014] noises also allow spectral control, but may introduce, generally marginal, interferences due to interpolation function w_k and the regularity of the $\mathbf{p}_k$.

Figures 1 to 4 illustrate how two key parameters affect the visual quality of solid Gabor and wave noises. For Gabor noise, the number of kernels (n) impacts both quality and framerates, while lattice size (given as the ratio in relation to the unit) primarily influences quality. Similarly, for wave noise, the number of directions (n) affects both quality and framerates, while average slice thickness (given as the ratio in relation to the unit) predominantly affects quality. The top left is the reference noise 3D texture computed by inverse 3D FFT. It is a filtered white noise, where the frequency band is shown as a bar chart.

Though framerates are provided as a general performance indicator in this Figure, they may vary during interactive manipulations. We refer to the paper's performance table for a more objective performance analysis. The Figures mainly illustre the effect of these parameters on quality.

Figure 5 extends the analysis with a quantitative 2D spectral accuracy evaluation of wave noise, showing error trends with respect to direction count and slice thickness. Errors are L2 norms averaged over non-zero frequencies. The reference (computed by inverse FFT) is shown top-left, and average PSDs were estimated from 16 random patches, as wave noise is unbounded by design.

2 Additional results

In this section we show additional results concerning transitions between wave noises, more anisotropic, non-Gaussian and cellular results, and more results concerning applications to material and color texture modeling and animation.

2.1 Transition results

Our noise relies on parameters that can be linearly interpolated to create smooth and consistent transitions, as illustrated in Figure 6. We show transitions of scale, transitions from Gaussian to ridged and Phasor noises, a transition from curl noise to cellular noise and transitions concerning waves with different frequency contents.

Transitions are limited to real-valued parameters and require the use of consistent wave combination operators. For instance, smoothly transitioning between our cellular structure, which relies on the min operator, and Gaussian noise, which uses a sum, presents a challenge not addressed in the paper.

2.2 More results

Our model combines waves oriented in random directions. When certain directions are favoured, it creates anisotropic patterns. More examples than presented in the paper are shown in Figure 7

Our style transfer functions are generated from PBR material maps. We generate them using a cube map, which is then projected on a sphere for indexation according to the normal. The same wave noise can produce diverse materials by applying different styles. Figure 8 demonstrates this with a wave noise, resembling a marble structure. In this example, the style transfer functions combine two styles, as shown on the left.

Figure 9 illustrates the use of wave noise to guide an optimization-based texture synthesis approach, as presented in [Guehl et al. 2020] and named by the authors "semi-procedural" textures. Our approach likewise allows us to define such semi-procedural textures.

Figure 10 illustrates material animation on four examples, completing the results presented in the paper and video : the glowing stone cooling down (with additional time steps), moss slowly growing on a rock (with additional time steps), some frames of snow melting and a climbing plant, spreading over a tree trunk, as presented in the video.

3 Model parameters

This section details wave noise parameters. It mainly consists of a collection of screenshots illustrating how these parameters can be adjusted through our GUI and affect visual results. The Table below summarizes wave noise parameters.

Parameters of $\mathcal{N} : \mathbb{R}^3 \times \mathbb{R} \to \mathbb{C}$
$N_\omega = N_\varphi N_\theta$ (number of directions)
$T(x) : [0,1] \to \mathbb{R}^2$ (precomputed table)
Op $\in$ { +, min, stack} (wave combination)
W (slice thickness)
F (normalization factor)
v (wave speed)

Figure 11 is a crop of a GUI, which proposes additional parameters and controls that we grouped as follows:

Spatial Transformation (Figure 12). These parameters affect 3D noise position and scale through translation (X, Y, Z) and global zoom factor. We remind the reader that procedural wave noise is unbounded in space, but for ease of use we limited the ranges in the GUI. These transormations are applied to $\mathbf{x}$ before calling $\mathcal{N}(\mathbf{x}, t)$.

Quality versus Framerates. Figure 13 shows how setting the slice thickness (parameter W), and the number of directions N_ω affect the results. In the GUI, we reduced to a single value, called $NDir$, the actual numbers N_φ and N_θ, by fixing $N_\theta = 4$ and $N_\varphi = NDir/4$.

Frequency Range (Figure 14). The frequency bounds (FreqMin, FreqMax) allow users to define a global band-pass filter. Outside this range, amplitudes are null. These parameters are used to precompute T. It affects only Gaussian noises (isotropic and anisotropic). The parameter *anisotropy spread* affects anisotropic noises, as described below.

Waveforms and Noise types allow users to choose from various preset waveforms, wave combinations (parameter Op), and degrees of anisotropy:

- Figure 15 illustrates isotropic Gaussian noises characterized by amplitude variations within the previous range [FreqMin, FreqMax]. We display the implemented amplitude variation presets (as bar charts), along with the corresponding real part of the precomputed complex-valued wave stored in the table T. For these noises, Op = + (waves are added).

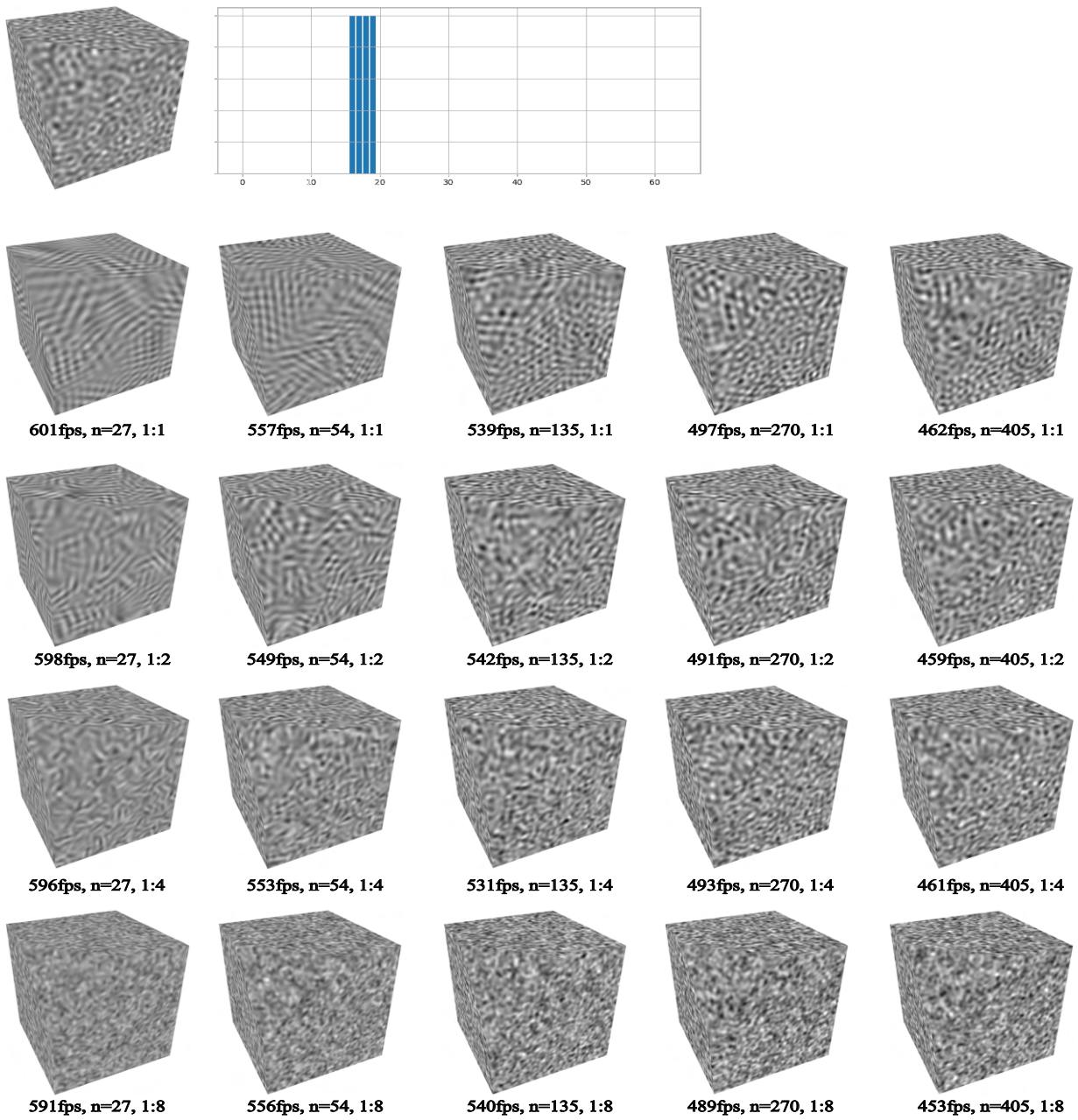

Fig. 1. Evaluating Gabor sparse convolution noise according to total number of kernels and grid size. Top is reference obtained by inverse 3D FFT. PSD is isotropic. The bar chart shows energy variation according to frequency. It corresponds to a perfectly band-pass filtered white noise.

- Figure 16 illustrates anisotropic Gaussian noises, also characterized by amplitude variations within the previous range `[FreqMin, FreqMax]`. Here, we implemented two kinds of anisotropic noises: one-directional or two-directional. The distinction from the previous isotropic noises lies in how wave directions are randomly selected. In the one-directional anisotropic case:
$(\alpha, \beta) := \left(\alpha_0 + \frac{\texttt{spread}}{4} \cdot \frac{\pi(i+\texttt{rnd}())}{N_\varphi+1}, \arccos\left(\cos(\beta_0) + \frac{\texttt{spread}}{4}\frac{j+\texttt{rnd}()}{N_\theta+1}\right)\right)$,
(α_0, β_0) being the main anisotropy direction. For the two-directional case, we define a second direction (α_1, β_1), and equally distribute the *NDir* randomly drawn directions over these two directions. The parameter *anisotropy spread* corresponds to the value of `spread` in the previous formula.The smaller its value the closer randomly drawn directions match (α_0, β_0).

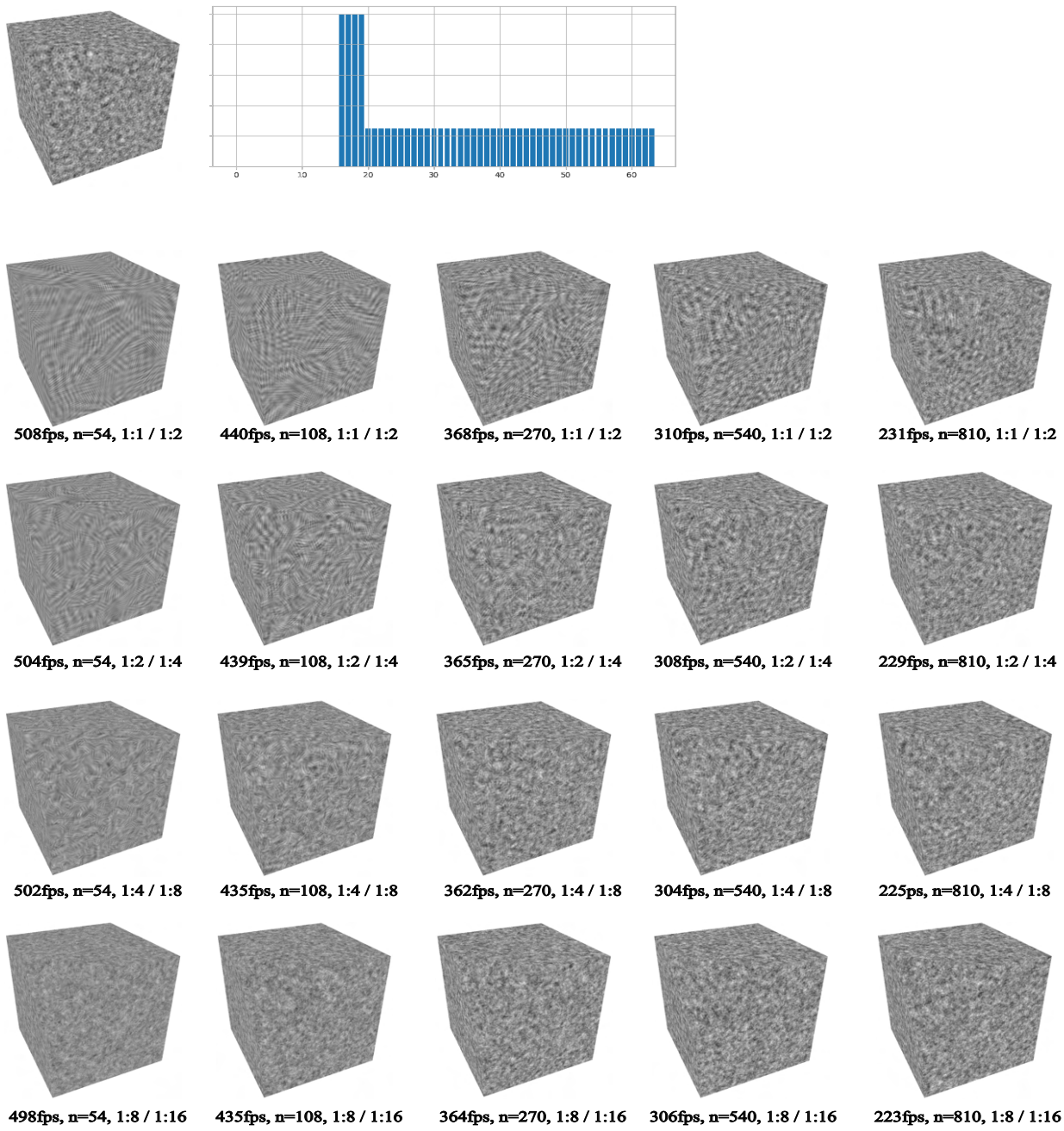

Fig. 2. Evaluating Gabor sparse convolution noise according to number of kernels and grid size. Top is reference obtained by inverse 3D FFT. PSD is isotropic with two distinct energy steps. For best performance, we used two grids with different sizes separating low and high frequency components of the noise.

- Figure 17 shows the preset collection of waveforms that we used to generate examples of non-Gaussian noises. These noises also use a sum of waves (`Op` is set to +). These waves are characterized by spikes whose thickness can be modulated with a parameter `nongauss wave sharpness`.
- Figure 18 shows cellular noises controlled by subdivision probability and maximum recursion depth, as available through

Cellular Noise Settings. These noises do not add waves, but use the *min* and *stack* (for random polytopes drawing) combination operation.

Post-Processing and Display define the value that is rendered (real, imaginary, modulus or argument), scaled by `contrast`, which we use to derive the normalization factor F.

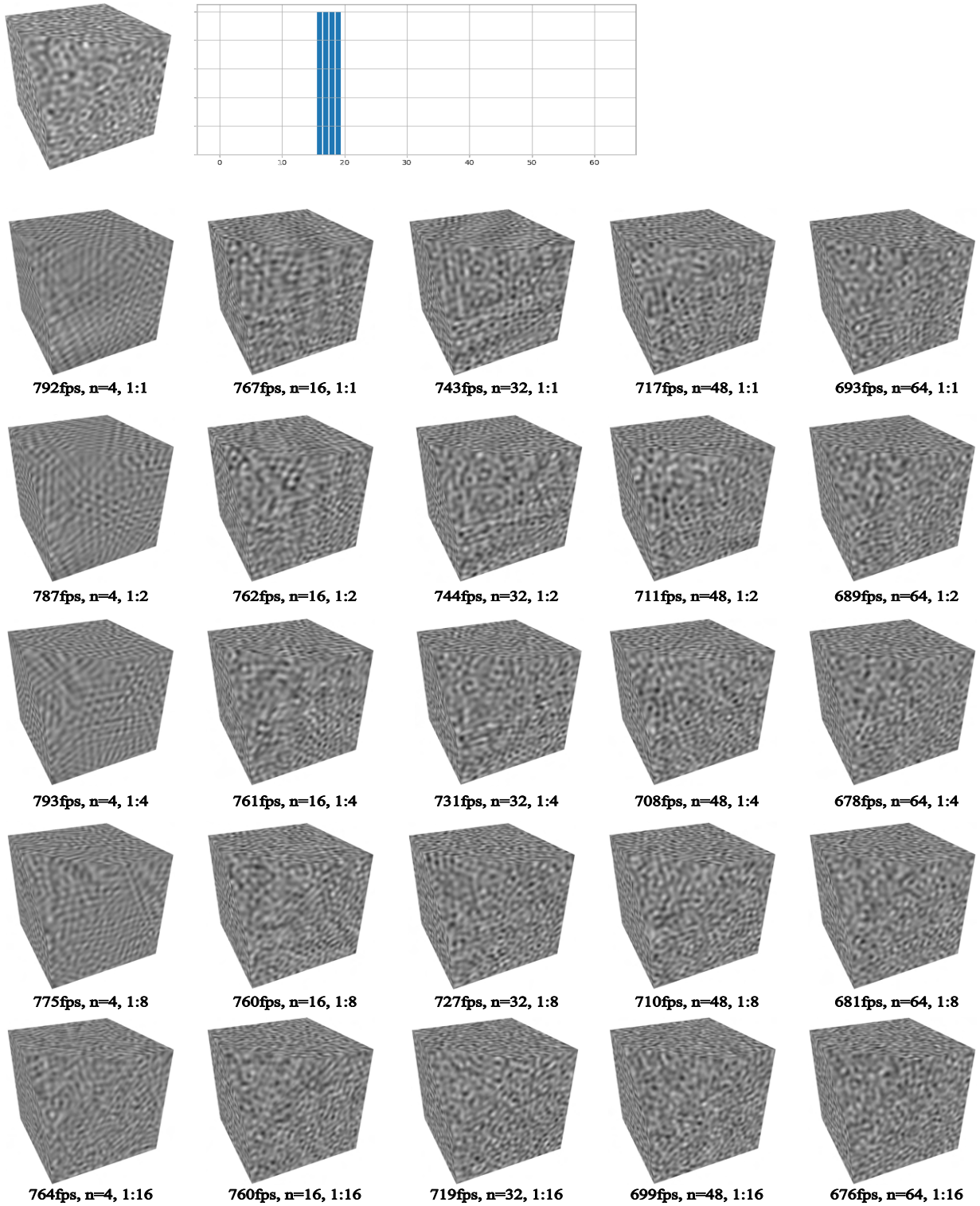

Fig. 3. Evaluating wave noise according to number of directions (n) and slice thickness. Top is reference obtained by inverse 3D FFT. PSD is isotropic. The bar chart shows energy variation according to frequency. It corresponds to a perfectly band-pass filtered white noise.

Temporal and Animation Controls add motion to the noise using a timeline position and wave speed (parameter v).

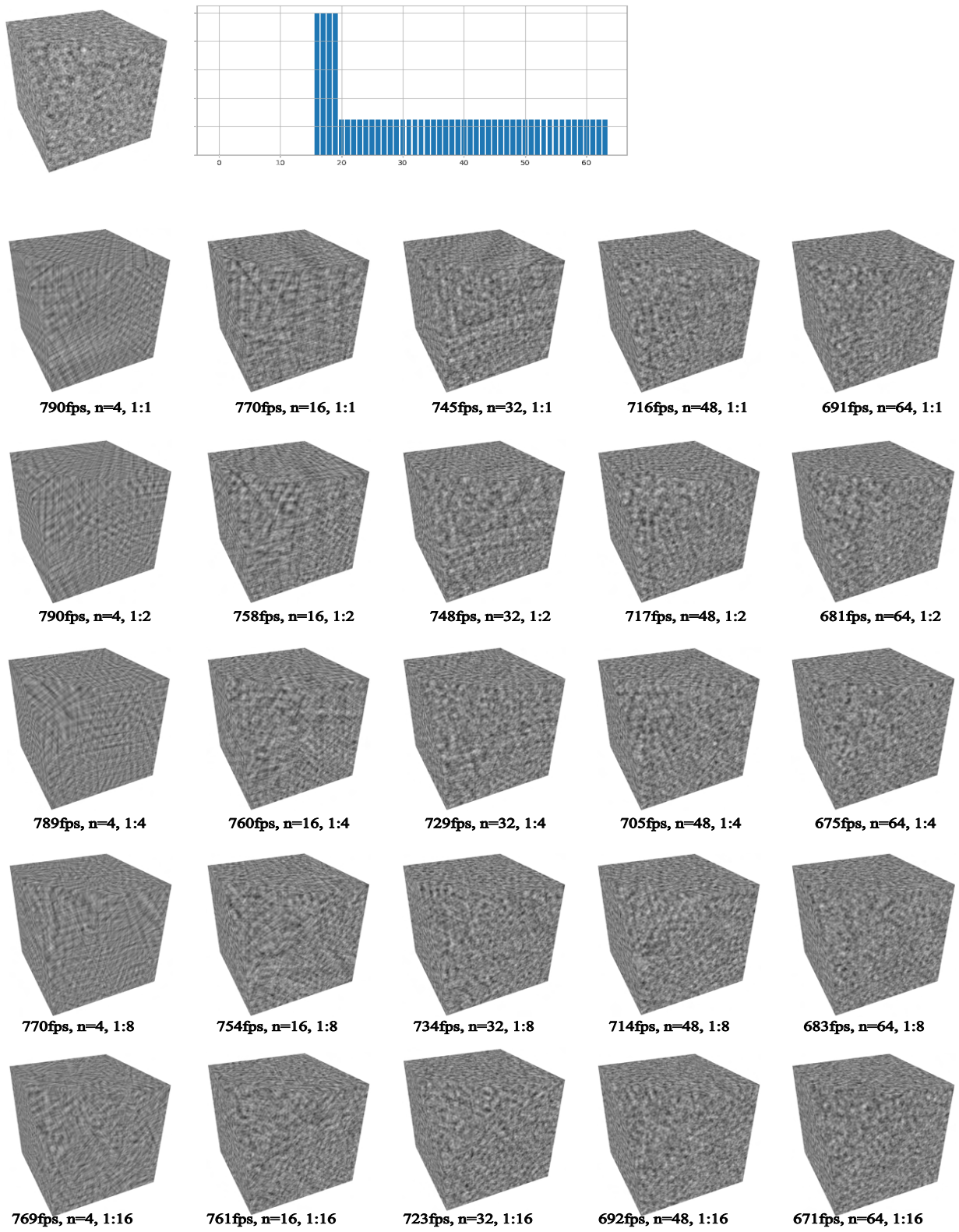

Fig. 4. Evaluating wave noise according to number of directions (n) and slice thickness. Top is reference obtained by inverse 3D FFT. PSD is isotropic with two distinct energy steps.

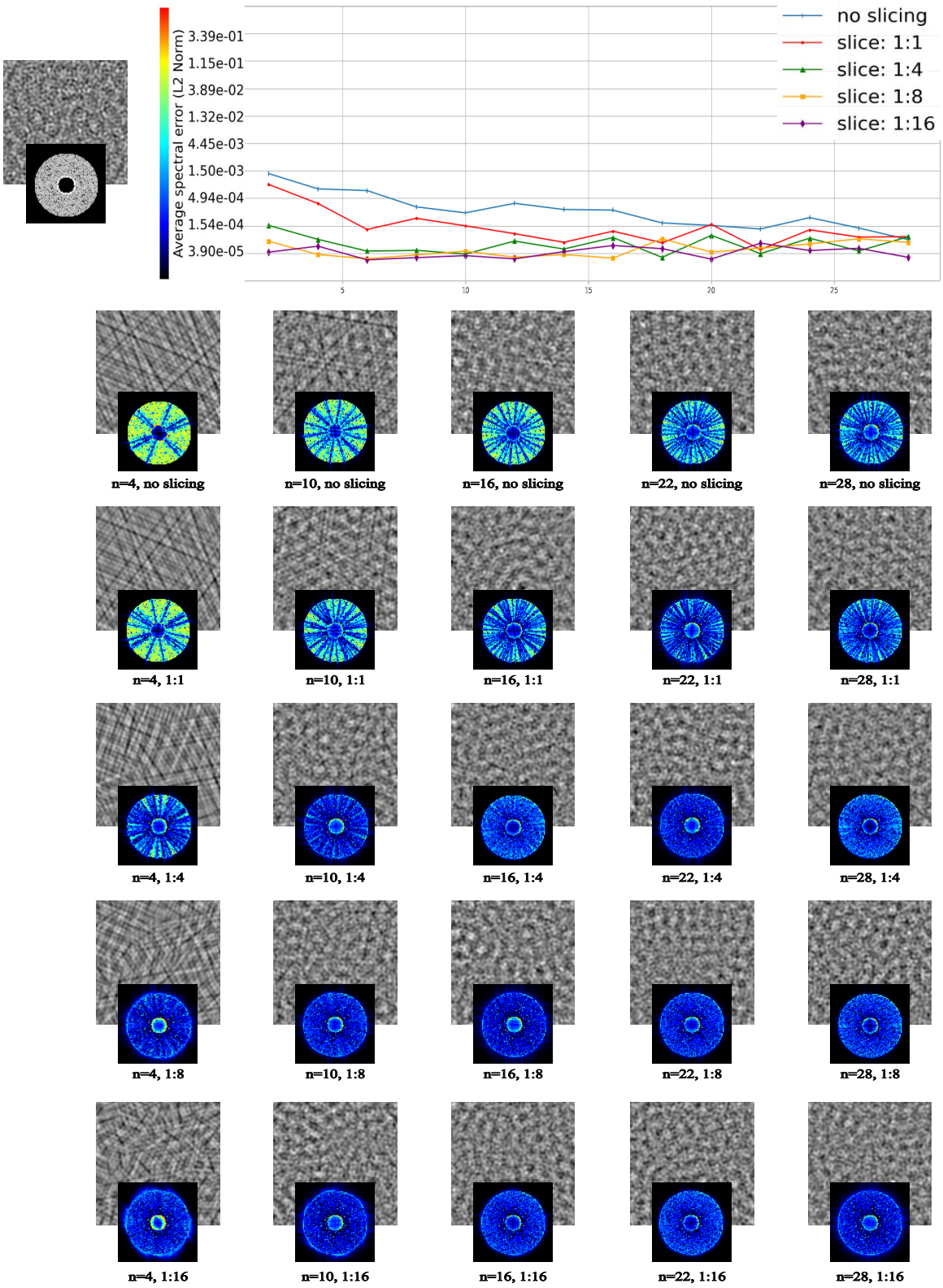

Fig. 5. Quantitative wave noise evaluation according to number of directions (n) and slice thickness. Top left is reference obtained by inverse 2D FFT.

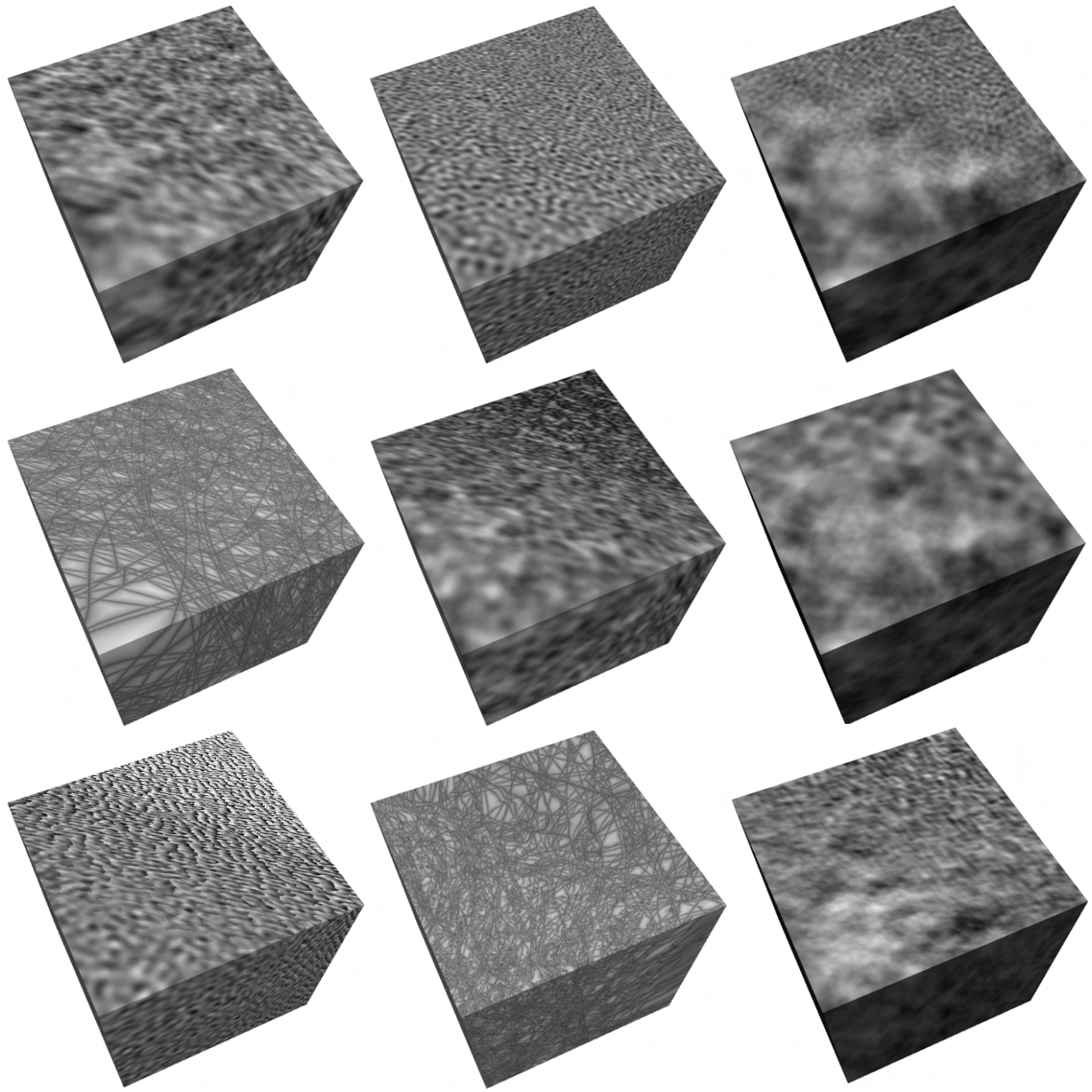

Fig. 6. Smooth transitions between wave noises from one cube corner to the opposite. The transition is for the entire volume, not only on the top plane.

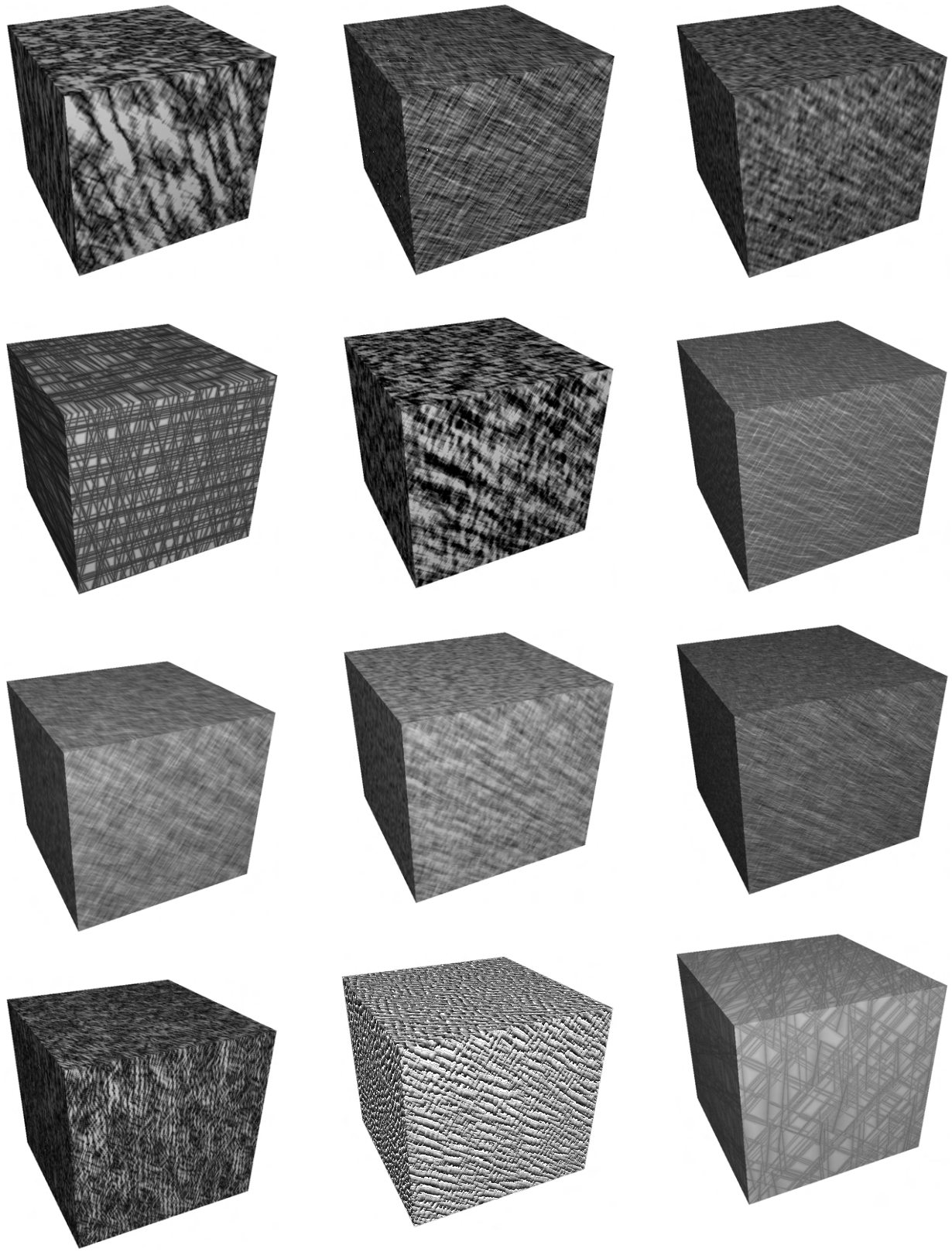

Fig. 7. Examples of anisotropic wave noises generated by favouring certain directions.

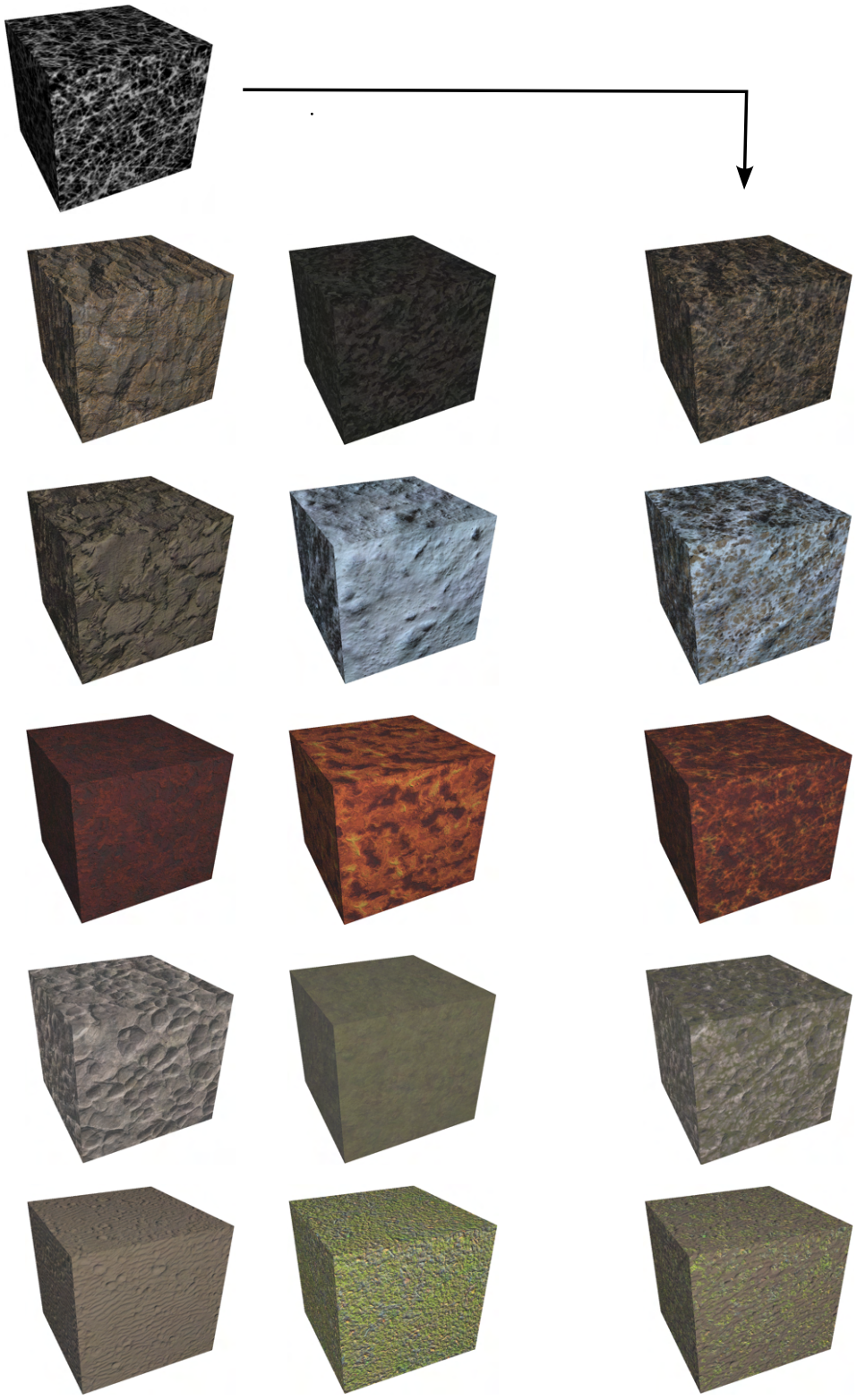

Fig. 8. Using non-Gaussian wave noise (top) to design materials (right column) from style transfer functions. On the left we applied the styles on cubes. No UV coordinates are required, neighter for applying the styles, nor for the wave noise, the latter being 3D.

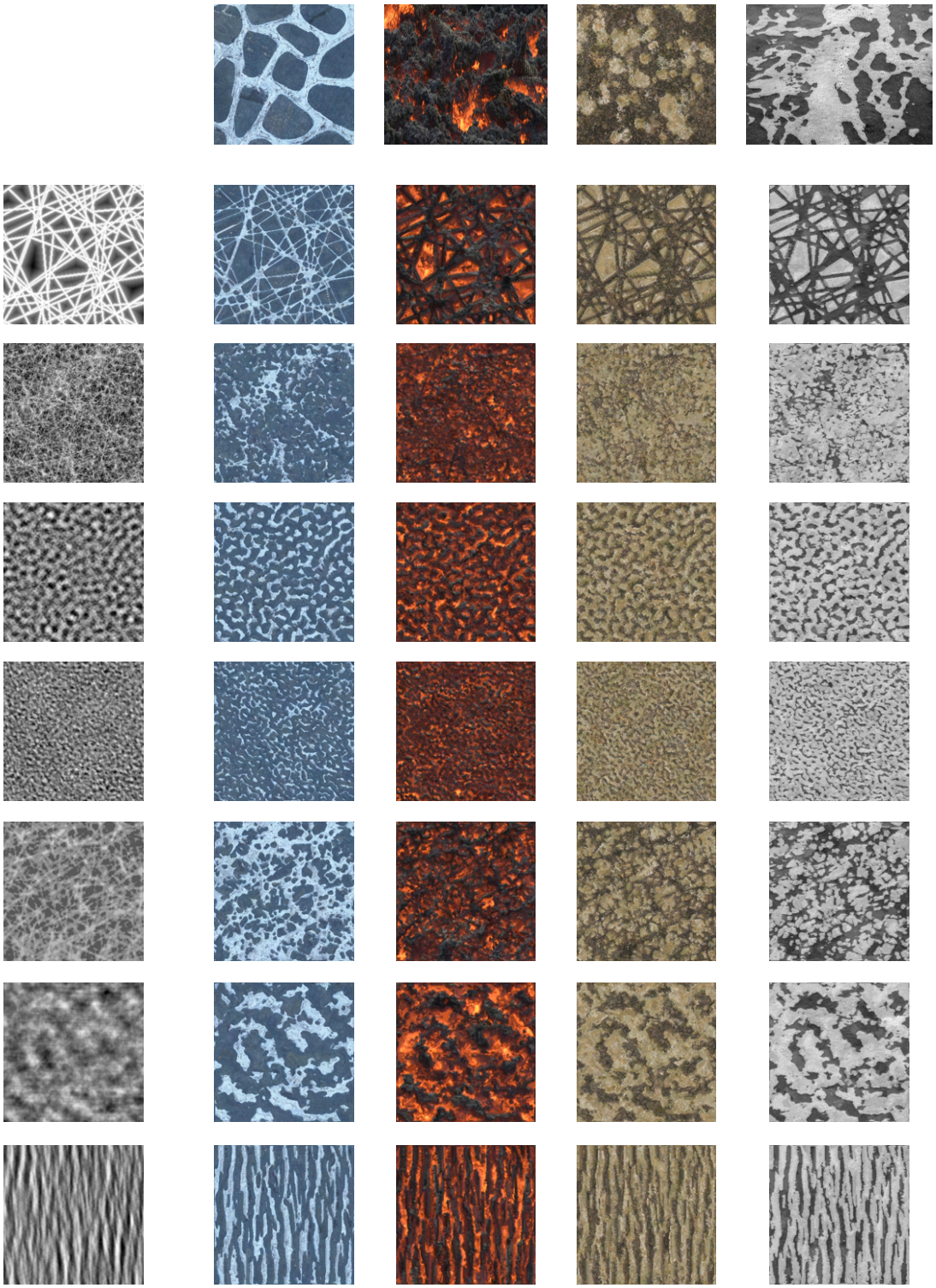

Fig. 9. Data-driven by-example color texture synthesis using wave noise (left column) as “guidance”. Input textures are shown on top row. The merging of a procedural model with data-driven synthesis was called “semi-procedural” texture synthesis in [Guehl et al. 2020]. The input textures on top are from Guehl et al’s database, as provided on their git repository.

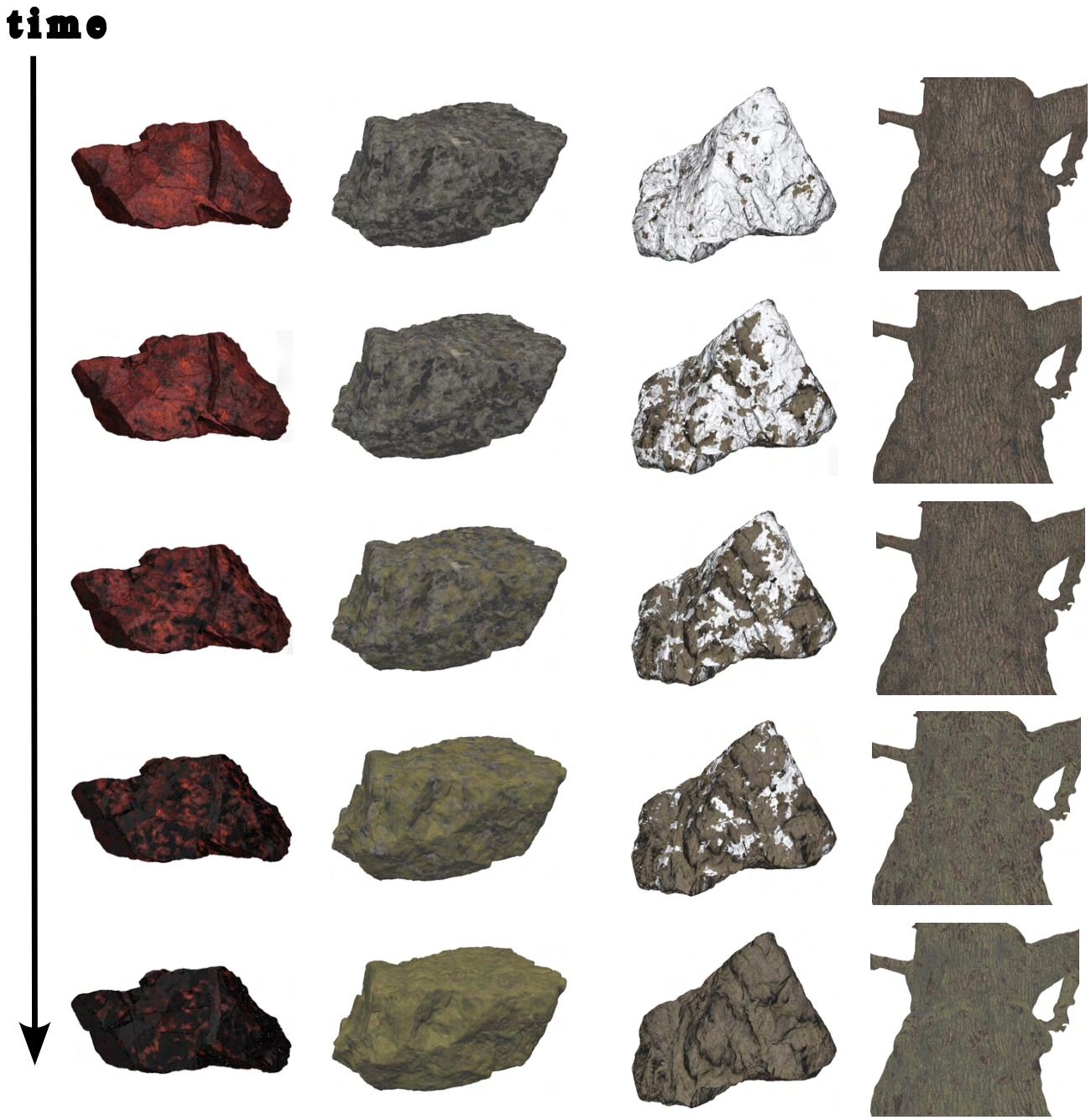

Fig. 10. Completing the paper's animation results with additional frames.

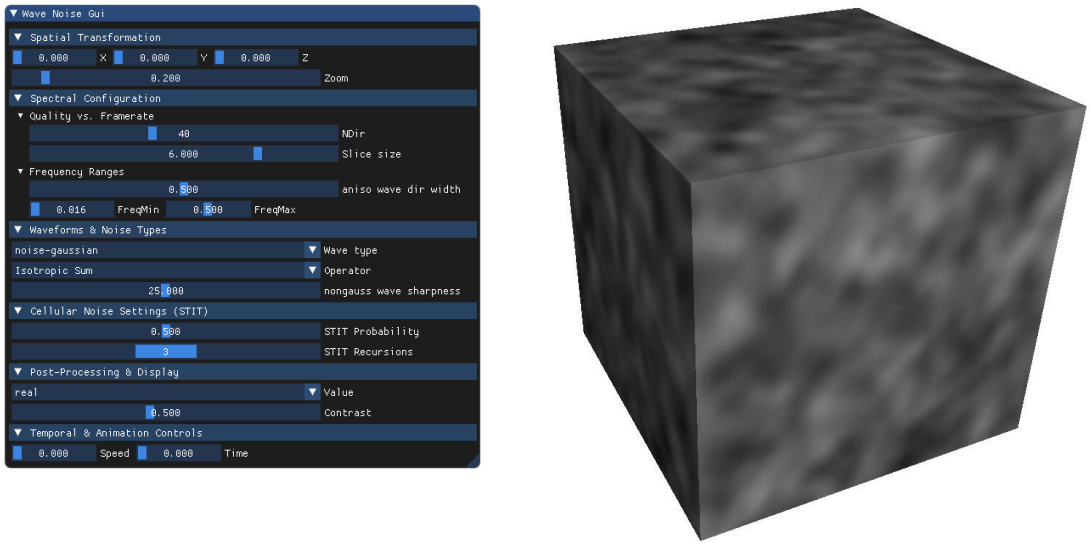

Fig. 11. Initial parameters and corresponding noise result.

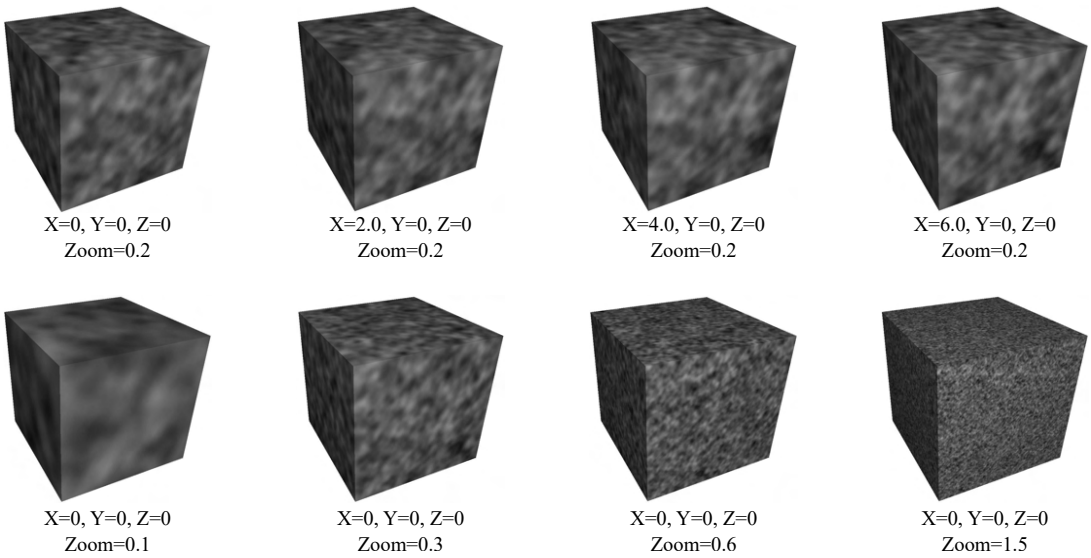

Fig. 12. **Spatial Transformation** controls. Top row: increasing *X* from left to right. Bottom row: increasing *Zoom* from left to right.

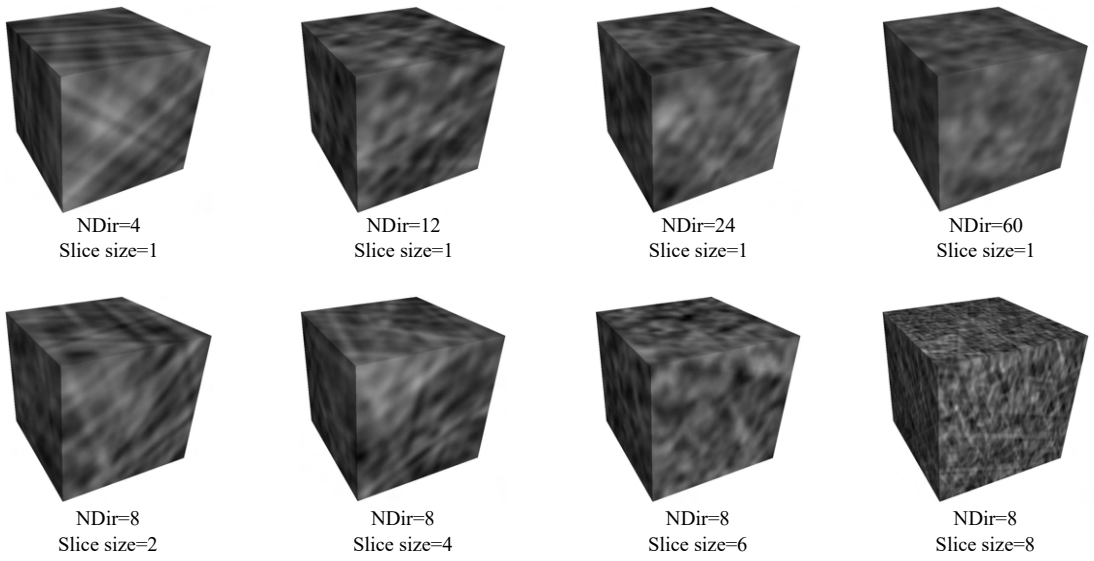

Fig. 13. **Quality versus Framerates** controls. Top row: increasing *NDir* from left to right. Bottom row: increasing *Slice size* from left to right.

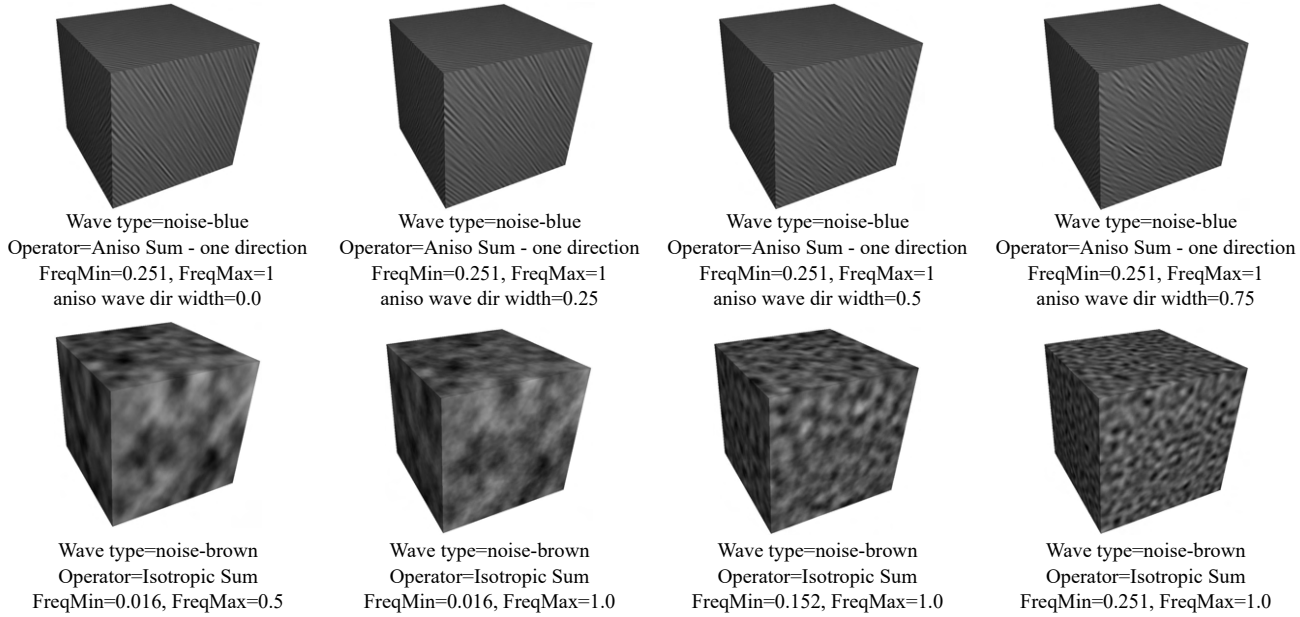

Fig. 14. **Frequency Range** controls. Top row: increasing *aniso wave dir width* from left to right. Bottom row: effect of varying `FreqMin` and `FreqMax` from left to right to control a band-pass filter.

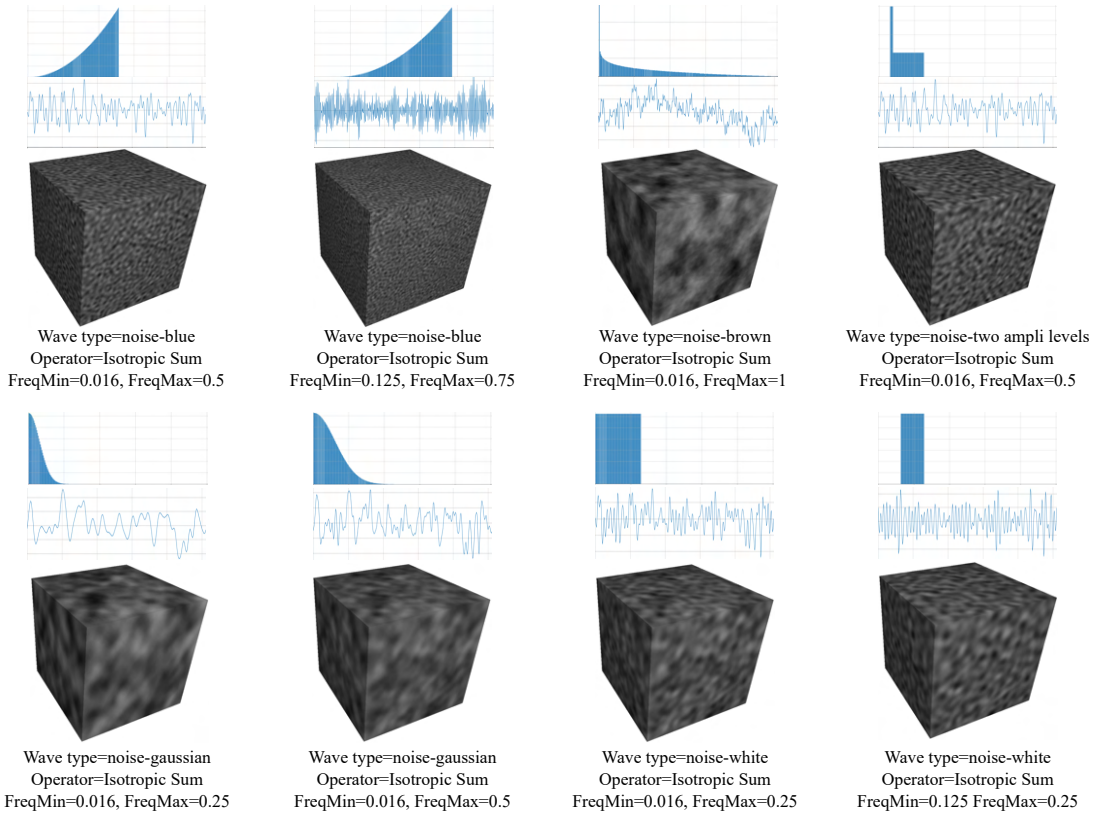

Fig. 15. **Waveforms and Noise types** controls: Isotropic Gaussian noises.

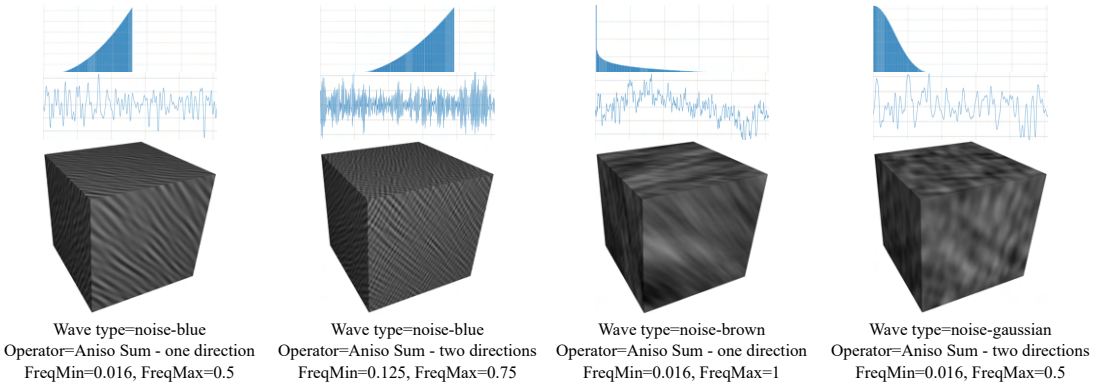

Fig. 16. **Waveforms and Noise types** controls: Anisotropic Gaussian noises.

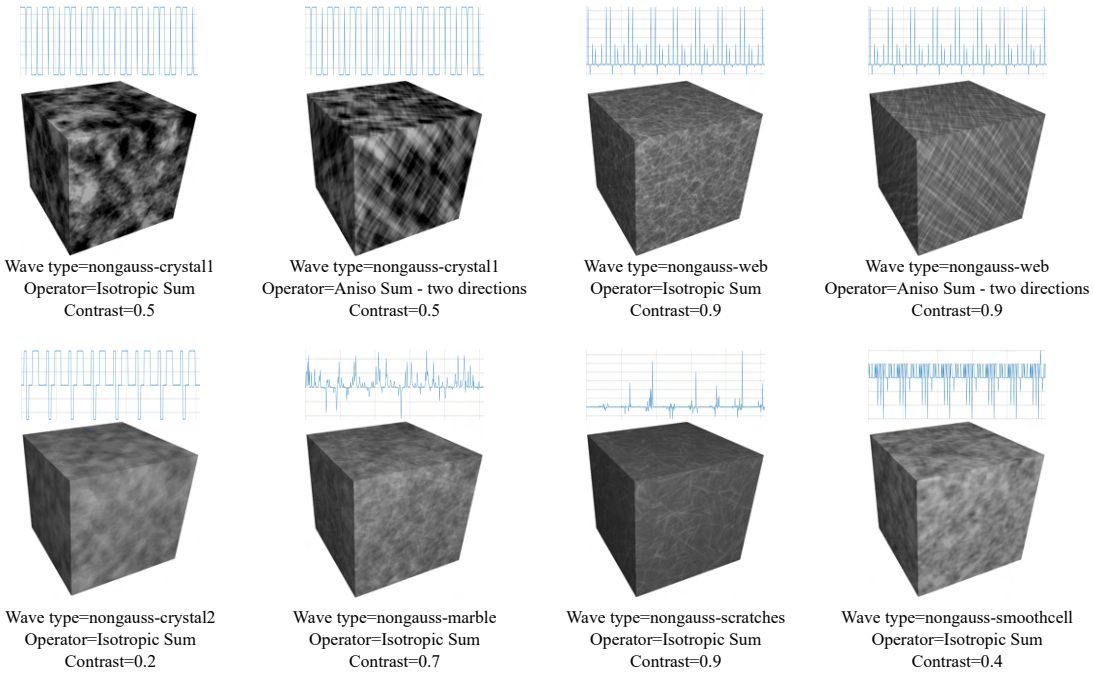

Fig. 17. **Waveforms and Noise types** controls: Non-Gaussian noises.

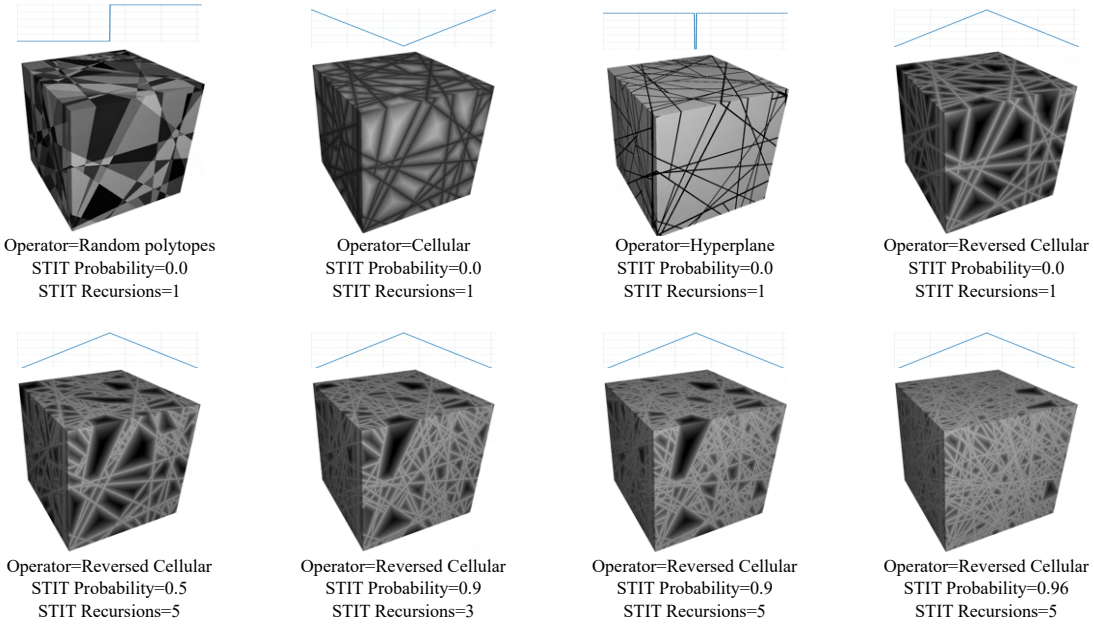

Fig. 18. **Waveforms and Noise types** controls: Cellular noises. Here, $NDir = 8$ for all examples. We show the corresponding waveforms. These are directly implemented as functions and not precomputed into table T, given their simplicity.

References

Arthur Cavalier, Guillaume Gilet, and Djamchid Ghazanfarpour. 2019. Local spot noise for procedural surface details synthesis. *Computers & Graphics* 85 (2019), 92–99.

Bruno Galerne, Ares Lagae, Sylvain Lefebvre, and George Drettakis. 2012. Gabor noise by example. *ACM Transactions on Graphics (ToG)* 31, 4 (2012), 1–9.

Bruno Galerne, Arthur Leclaire, and Lionel Moisan. 2017. Texton noise. In *Computer Graphics Forum*, Vol. 36. Wiley Online Library, 205–218.

Guillaume Gilet, Jean-Michel Dischler, and Djamchid Ghazanfarpour. 2012. Multiple kernels noise for improved procedural texturing. *The Visual Computer* 28 (2012), 679–689.

Guillaume Gilet, Basile Sauvage, Kenneth Vanhoey, Jean-Michel Dischler, and Djamchid Ghazanfarpour. 2014. Local random-phase noise for procedural texturing. *ACM Transactions on Graphics (ToG)* 33, 6 (2014), 1–11.

Pascal Guehl, Rémi Allegre, J-M Dischler, Bedrich Benes, and Eric Galin. 2020. Semi-Procedural Textures Using Point Process Texture Basis Functions. In *Computer Graphics Forum*, Vol. 39. Wiley Online Library, 159–171.

Eric Heitz and Fabrice Neyret. 2018. High-performance by-example noise using a histogram-preserving blending operator. *Proceedings of the ACM on Computer Graphics and Interactive Techniques* 1, 2 (2018), 1–25.

Ares Lagae, Sylvain Lefebvre, George Drettakis, and Philip Dutré. 2009. Procedural noise using sparse Gabor convolution. *ACM Transactions on Graphics (TOG)* 28, 3 (2009), 1–10.

John-Peter Lewis. 1984. Texture synthesis for digital painting. In *Proceedings of the 11th annual conference on Computer graphics and interactive techniques*. 245–252.

John-Peter Lewis. 1989. Algorithms for solid noise synthesis. In *Proceedings of the 16th annual conference on Computer graphics and interactive techniques*. 263–270.

Ken Perlin. 1985. An image synthesizer. In *Proceedings of the 12th Annual Conference on Computer Graphics and Interactive Techniques (SIGGRAPH '85)*. Association for Computing Machinery, New York, NY, USA, 287–296. doi:10.1145/325334.325247

Ken Perlin. 2001. Noise Hardware. In *Real-Time Shading SIGGRAPH Course Notes*, M. Olano (Ed.). Chapter 9.

Jarke J Van Wijk. 1991. Spot noise texture synthesis for data visualization. In *Proceedings of the 18th annual conference on Computer graphics and interactive techniques*. 309–318.