Quad-optimized Low Discrepancy sequences: *Supplementary material*

Victor Ostromoukhov
Univ Lyon 1, CNRS, INSA Lyon
France

Nicolas Bonneel
CNRS, Univ Lyon 1, INSA Lyon
France

David Coeurjolly
CNRS, Univ Lyon 1, INSA Lyon
France

Jean-Claude Iehl
Univ Lyon 1, CNRS, INSA Lyon
France

ACM Reference Format:
Victor Ostromoukhov, Nicolas Bonneel, David Coeurjolly, and Jean-Claude Iehl. 2024. Quad-optimized Low Discrepancy sequences: *Supplementary material*. In *Special Interest Group on Computer Graphics and Interactive Techniques Conference Conference Papers '24 (SIGGRAPH Conference Papers '24), July 27-August 1, 2024, Denver, CO, USA*. ACM, New York, NY, USA, 11 pages. https://doi.org/10.1145/3641519.3657431

1 ADDITIONAL RESULTS IN 2D

In Figures 1, 2 and 3, we provide generalized l_2 discrepancy measures and integrations errors (both for Gausisan and Heaviside functions), for consecutive 2d pairs of dimensions. In Fig. 4, we examine the discrepancy behavior for sample counts where no guarantee is provided: within octaves and outside of the first $3^{10} \approx 59k$ samples for a selected pair of dimensions. In Fig. 5, we further examine discrepancy and integration error for an unoptimized pair of projections (pair $(7, 11)$, counting from 0) that apparently produces block patterns for 243 samples (Fig.4(f) of the main paper ; for 243 samples, $t = 2$). Despite this apparent non-uniformity, generalized l_2 discrepancy remains lower than white noise, ZeroTwo and Padded Sobol, and integration is in most cases better as well.

2 ADDITIONAL RESULTS IN 4D

In Figures 6 and 7 we provide generalized l_2 discrepancy measures and integrations errors (both for Gausisan and Heaviside functions), for consecutive 4d quadruples of dimensions.

3 ADDITIONAL RESULTS IN HIGHER DIMENSIONS

In Figures 8, 9 and 10 we provide generalized l_2 discrepancy measures in dimension 6, 8 and 12.

SIGGRAPH Conference Papers '24, July 27-August 1, 2024, Denver, CO, USA

ACM ISBN 979-8-4007-0525-0/24/07
https://doi.org/10.1145/3641519.3657431

4 RENDERING RESULTS

In Figures 11, 12 and 13, we provide rendered images for various per-pixel sample counts. MSE convergence graphs are given in Fig. 5 of the main document.

5 FASTER POINT GENERATION

Several computations can be simplified using 2 properties : Gray-code ordering and rewriting the matrix-vector product as a linear combination of columns of the matrix.

5.1 Rewriting matrix-vector product

The standard matrix vector product, for example on a 3×3 matrix, can be written as :

$$\begin{bmatrix} a_0 & a_1 & a_2 \\ b_0 & b_1 & b_2 \\ c_0 & c_1 & c_2 \end{bmatrix} \begin{bmatrix} i_0 \\ i_1 \\ i_2 \end{bmatrix} = \begin{bmatrix} a_0 i_0 + a_1 i_1 + a_2 i_2 \\ b_0 i_0 + b_1 i_1 + b_2 i_2 \\ c_0 i_0 + c_1 i_1 + c_2 i_2 \end{bmatrix}$$

and rewritten as a linear combination of columns of the matrix :

$$i_0 \begin{bmatrix} a_0 \\ b_0 \\ c_0 \end{bmatrix} + i_1 \begin{bmatrix} a_1 \\ b_1 \\ c_1 \end{bmatrix} + i_2 \begin{bmatrix} a_2 \\ b_2 \\ c_2 \end{bmatrix}$$

5.2 Gray-code ordering

The classical Sobol' point generation computes a matrix-vector product. The vector components are the digits of the index of the Sobol' point. Digits of two consecutive indices have no simple relation, but digits of two consecutive Gray codes differ in at most one place. Producing the next Sobol' point in Gray code order requires adding or subtracting a single column of the Sobol' matrix to the previous point. For instance, given a point whose Gray code order only has $i_0 \neq 0$ and $i_1 \neq 0$, but $i_2 = 0$, one of its coordinates p can be computed as $p = i_0 \begin{bmatrix} a_0 \\ b_0 \\ c_0 \end{bmatrix} + i_1 \begin{bmatrix} a_1 \\ b_1 \\ c_1 \end{bmatrix}$ using the rewritten matrix-vector product. If the Gray code order of the next point p' in Gray code order only alters i_2 such that $i_2 \neq 0$, then the corresponding coordinate can be efficiently computed as $p' = p + i_2 \begin{bmatrix} a_2 \\ b_2 \\ c_2 \end{bmatrix}$.

Gray codes have been generalized to arbitrary bases in [Guan 1998] and these properties hold in base 3.

6 OWEN SCRAMBLING IN BASE-3

6.1 Method

Owen scrambling, or nested unifom scrambling, can be compactly described with the following loop:

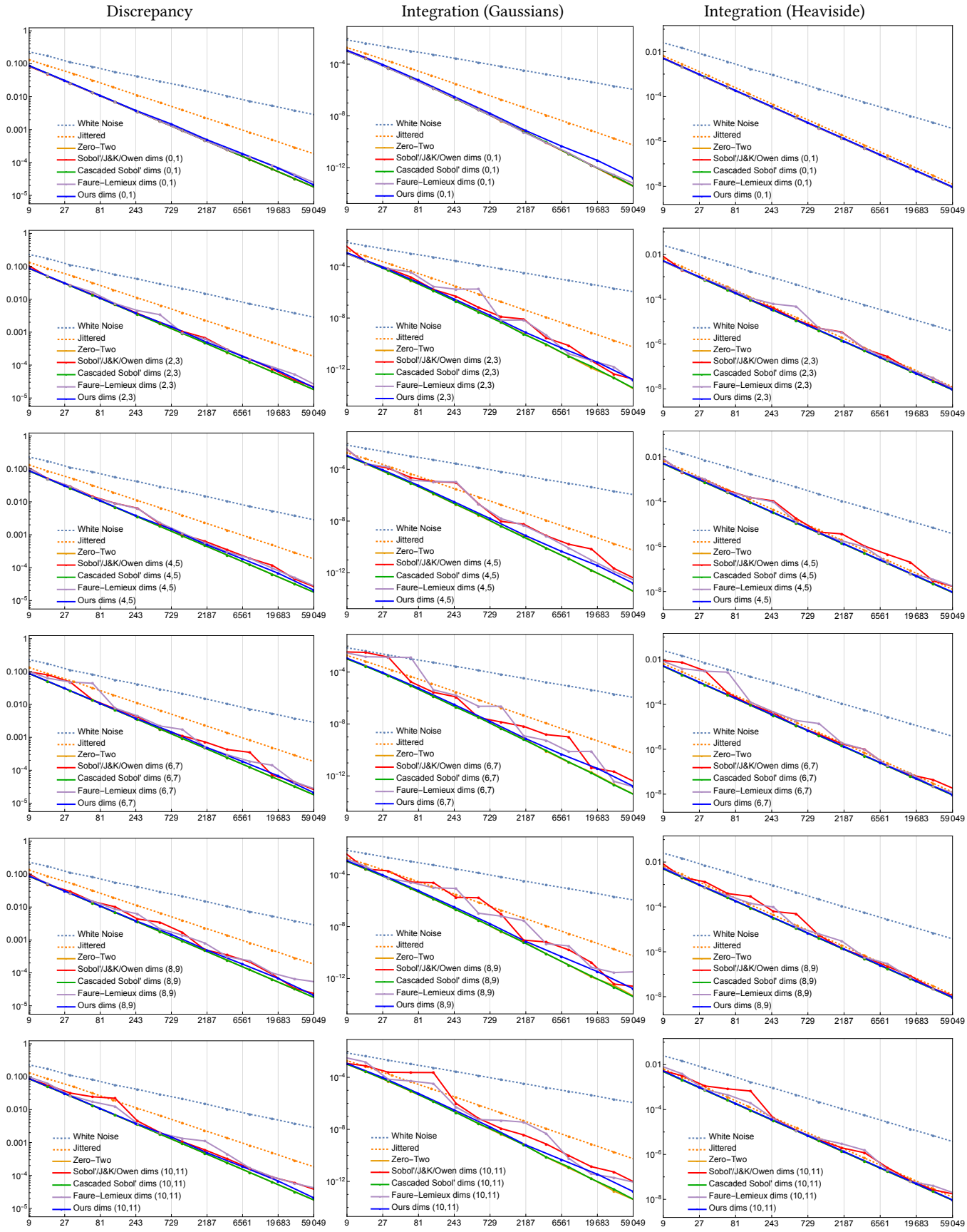

Figure 1: 2-d discrepancy and integration errors (MSE) for Gaussian and Heaviside functions for all consecutive pairs in $(2i, 2i+1)_{i=0..5}$ against competitors (see paper).

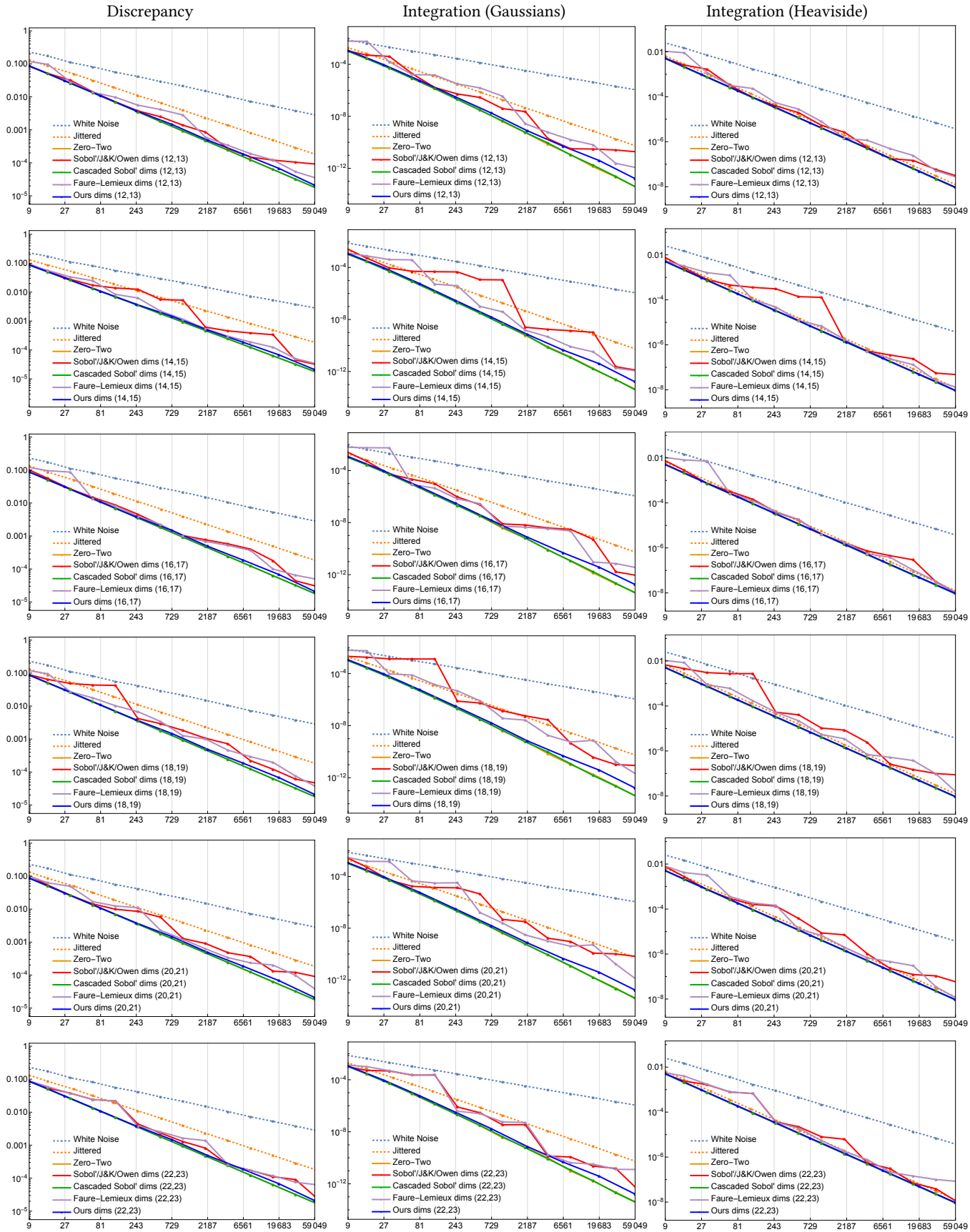

Figure 2: 2-d discrepancy and integration errors (MSE) for Gaussian and Heaviside functions for all consecutive pairs in $(2i, 2i+1)_{i=6..11}$ against competitors (see paper).

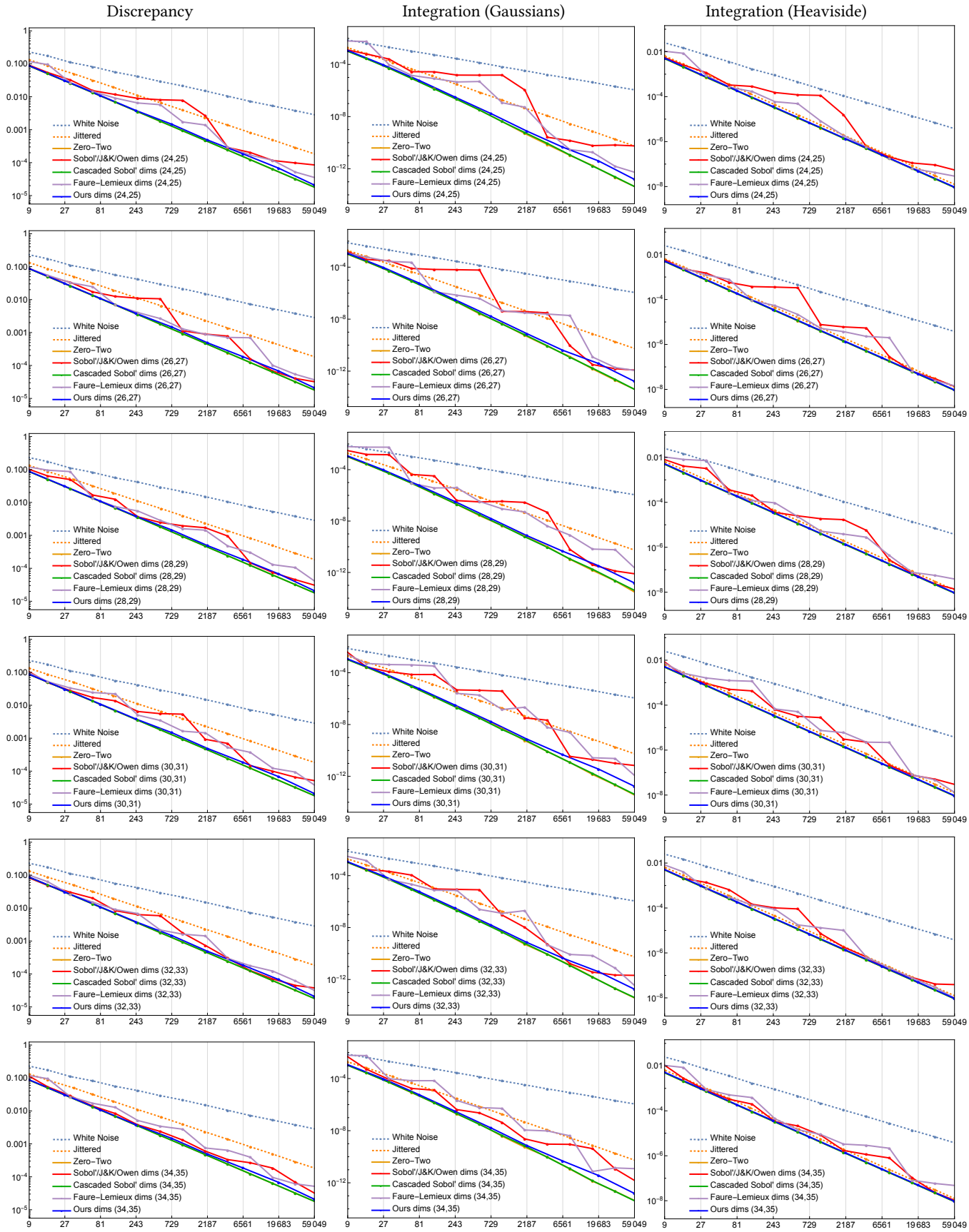

Figure 3: 2-d discrepancy and integration errors (MSE) for Gaussian and Heaviside functions for all consecutive pairs in $(2i, 2i+1)_{i=12..17}$ against competitors (see paper).

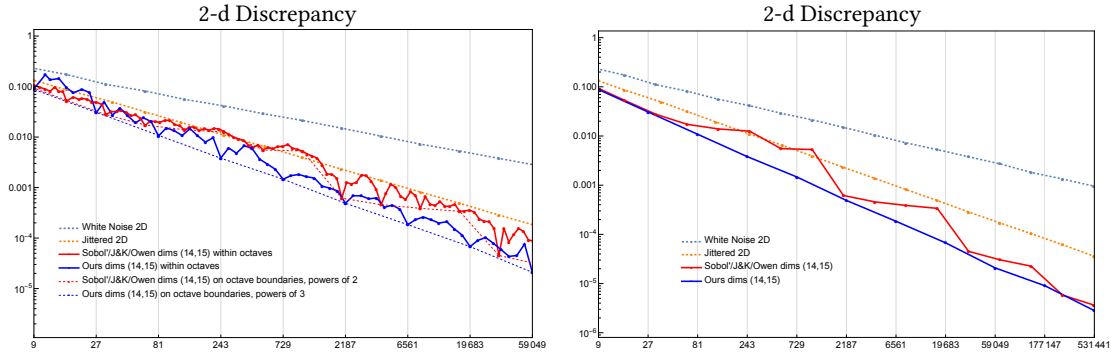

Figure 4: For a selected pair of dimensions $(14, 15)$, we examine the discrepancy for sample counts outside of the $(0, 2)$-progressive optimization. Left. We examine within octaves, at one third of each octave, for both Sobol' and our solution. For these sample counts, discrepancy is not guaranteed. Right. We examine above the ~59k samples, and see that the behavior seems to remain for 531k samples despite the relatively high degree of the polynomials involved (degrees 4 and 5).

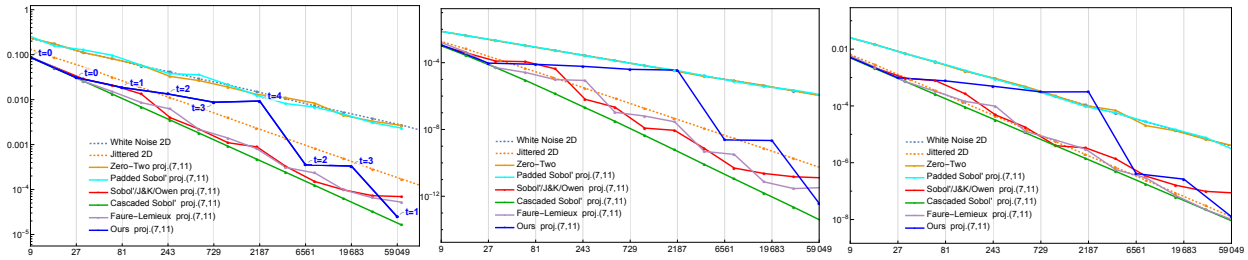

Figure 5: We investigate the pair of dimensions $(7, 11)$ which appears to have low uniformity for 273 samples (Fig.4(f) in the main paper) and structured block artifacts. Despite this apparent non-uniformity, it still generally performs better than random and the corresponding randomly paired dimensions of ZeroTwo and Padded Sobol'. For our sampler, the t value of each sample count is illustrated on the discrepancy graph, which correlates well with discrepancy.

```
scramble[6][3] ← {
{0, 1, 2}, {0, 2, 1}, {1, 0, 2},
{1, 2, 0}, {2, 0, 1}, {2, 1, 0}
}
▷ all random permutations of base-3 digits ◁
function OwenScramble(a : Sobol' point, m : Sobol' matrix size)
    result ← 0
    node_index ← 0
    ▷ start at the root node of the permutation tree ◁
    for i ← m − 1, . . . , 0 do
        ▷ all digits from most significant ◁
        digit ← a.digits[i]
        ▷ get a digit ◁
        s ← rng.index(node_index).sample_range(6)
        ▷ get a random permutation ◁
        result.digits[i] = scramble[s][digit]
        ▷ apply the permutation ◁
        node_index ← 3 × node_index + digit + 1
        ▷ continue walking the permutation tree ◁
    return result
```

This loop traverses a permutation tree. Each node of the permutation tree is numbered (variable *node_index*) and is associated with a term in a random sequence. Early implementations of Owen scrambling used to store the random sequence [Matousek 1998], but we use a counter-based (variable *rng*) random number generator [Salmon et al. 2011] to efficiently generate each node random permutation.

6.2 Results

In Fig. 14 we illustrate the effect of Owen scrambling in 2-d. We use an unoptimized pair of dimension that exhibits structured blocks of samples and voids (pair $(7, 11)$), and perform 25 realizations of Owen scrambling. Blocks remains, although at different locations, because Owen scrambling preserves the value of t (here, $t = 2$). By superimposing these 25 realizations, we illustrate the statistical uniformity: each block on average receives the same number of points.

For rendering, each pixel uses an independently scrambled sequence to decorrelate realizations. In Fig. 15, we use the same unoptimized pair of dimensions $(7, 11)$ to render a simple scene with direct lighting only in 2-d. We can see that even if structured blocks are present in each scrambled sequence, they do not manifest as structured artifacts in the image since blocks locations are decorrelated. In fact, even with this unoptimized sequence specifically

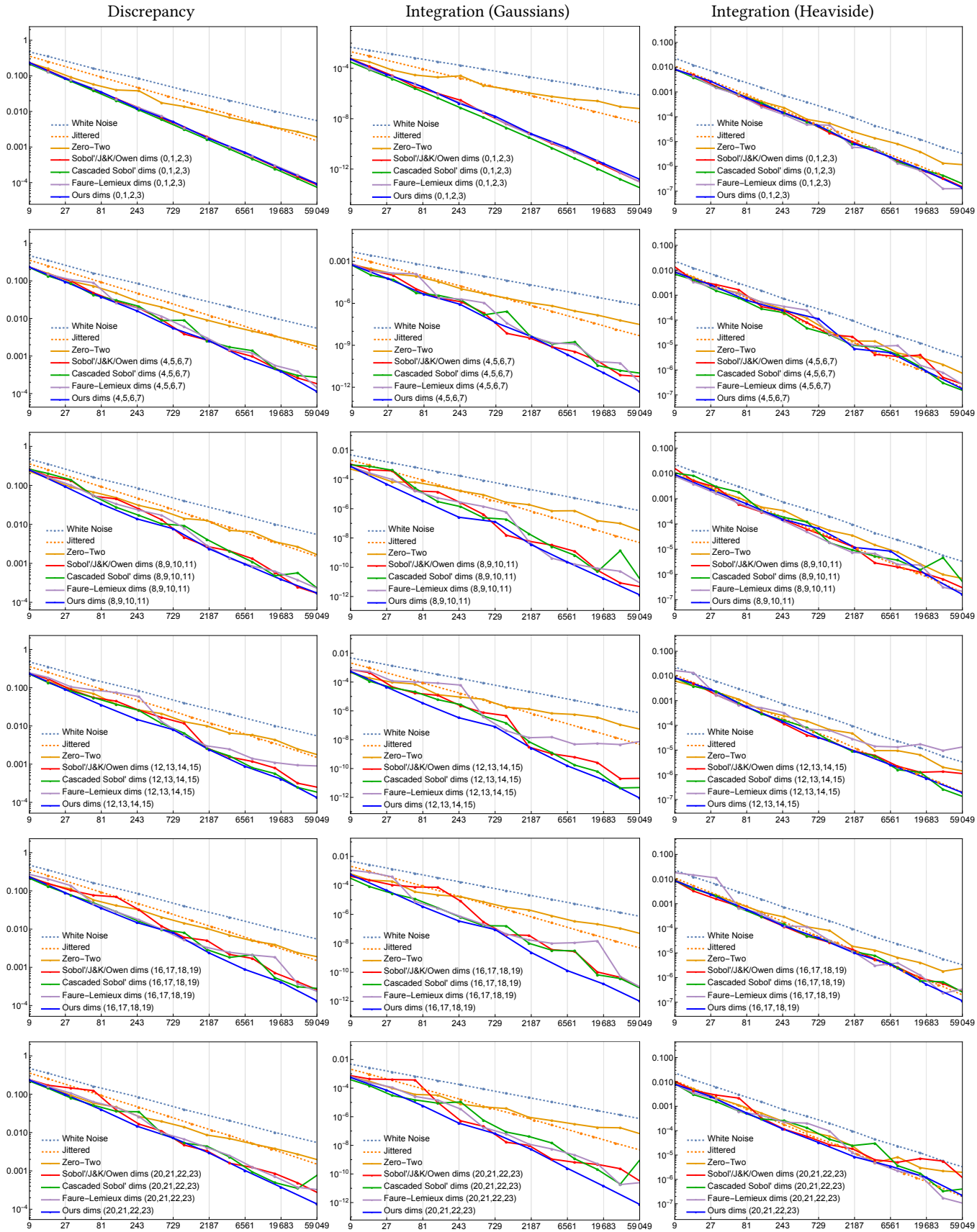

Figure 6: 4-d discrepancy and integration errors (MSE) for Gaussian and Heaviside functions for all consecutive quads in $(4i, 4i+1, 4i+2, 4i+3)_{i=0..5}$ against competitors (see paper).

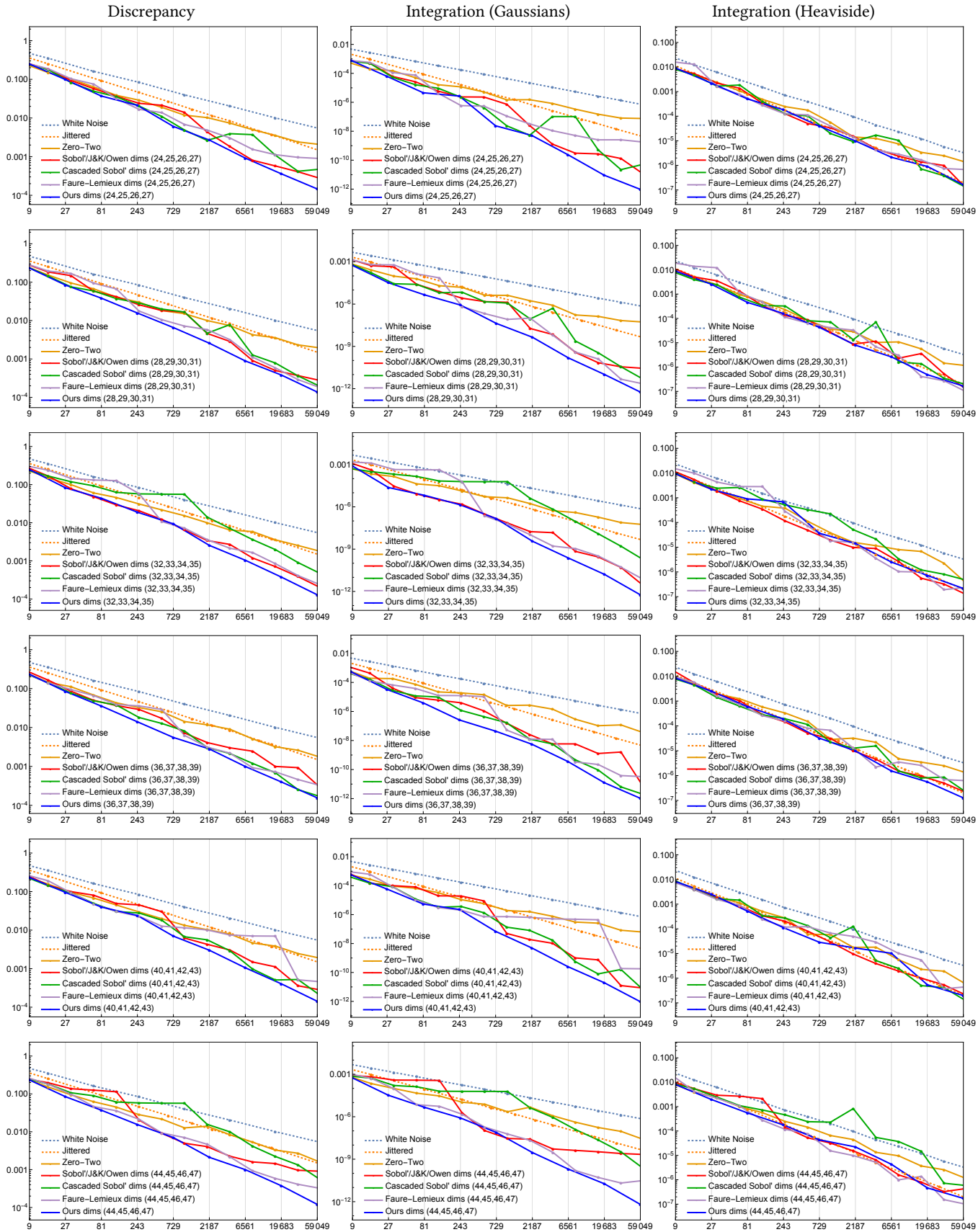

Figure 7: 4-d discrepancy and integration errors (MSE) for Gaussian and Heaviside functions for all consecutive quads in $(4i, 4i+1, 4i+2, 4i+3)_{i=6..11}$ against competitors (see paper).

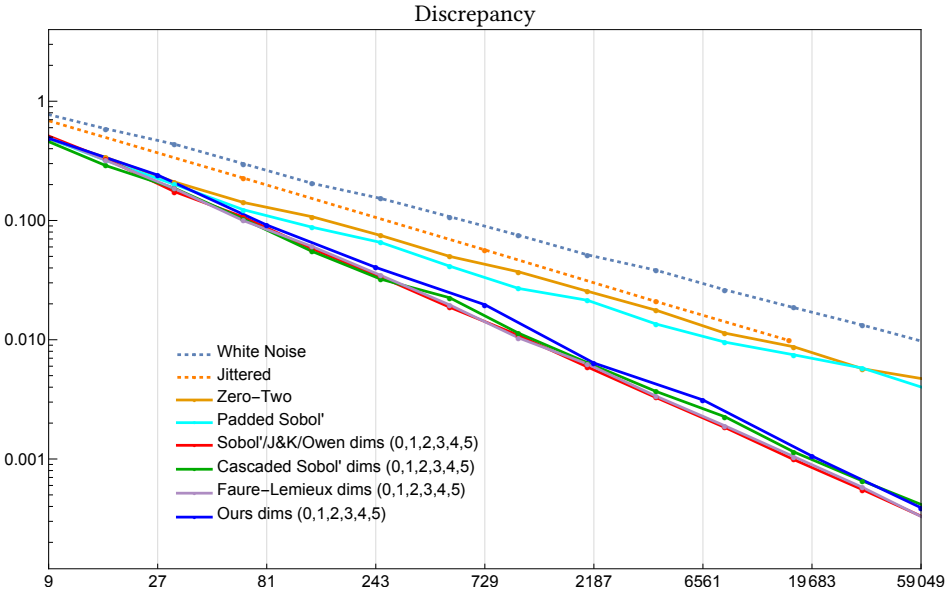

Figure 8: 6-d discrepancy of the first 6 dimensions of our sequence against competitors (see paper).

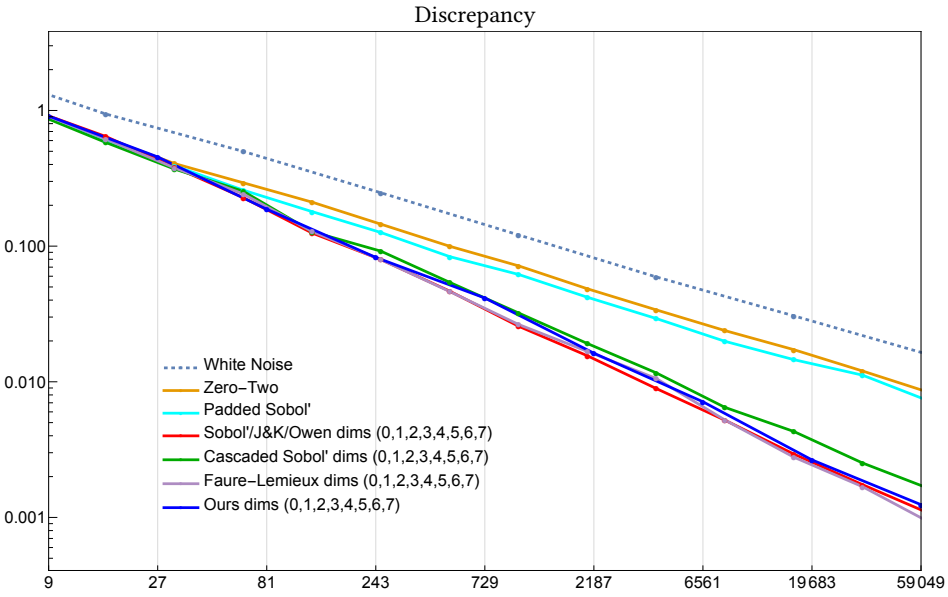

Figure 9: 8-d discrepancy of the first 8 dimensions of our sequence against competitors (see paper). It is worth noting that the slope of our 8-d curve (dark blue) is close to that of a typical 4-d low-discrepancy sequence (e.g., Sobol'/Owen or Faure-Lemieux), while the slope of Padded Sobol' and Zero-Two is close that White Noise. Padded Sobol' uses 4D quads of Sobol'/Owen of dimensions $(0, 1, 2, 3)$, with random mapping between quads. This introduces "white noise" component in this 8-d construction, even though the curve of 8-d Padded Sobol' lies below that of pure White Noise. The curve of 8-d Zero-Two approach can be explained similarly: this method uses pairs of dimensions such as (0,2)-seqences of excellent uniformity, which are linked randomly. Consequently, 8-d Zero-Two and Padded Sobol' behave similarly (the curve of Padded Sobol' is slightly better, but their slopes are very close).

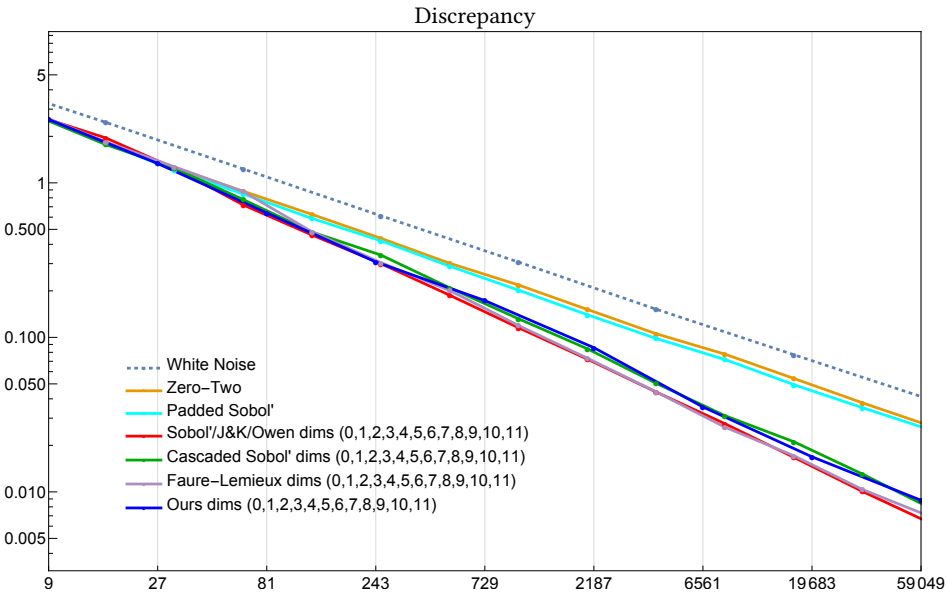

Figure 10: 12-d discrepancy of the first 12 dimensions of our sequence against competitors (see paper). The considerations for Padded Sobol' and Zero-Two described above are applicable here as well. Consequently, 12-d Zero-Two and Padded Sobol' behave similarly; the slope of both curves is close to that of White Noise.

Figure 11: 6-d renderings of a simple scene with several materials and large light sources. Direct lighting with a multi-sample MIS estimator.

selected for its lower uniformity, noise is reduced and convergence improved over white noise.

REFERENCES

Dah-Jyh Guan. 1998. Generalized Gray codes with applications. In *PROC NATL SCI COUNC REPUB CHINA PART A PHYS SCI ENG*, Vol. 22. 841–848.

Jiri Matousek. 1998. On the L2-Discrepancy for Anchored Boxes. *Journal of Complexity* 14, 4 (1998), 527–556.

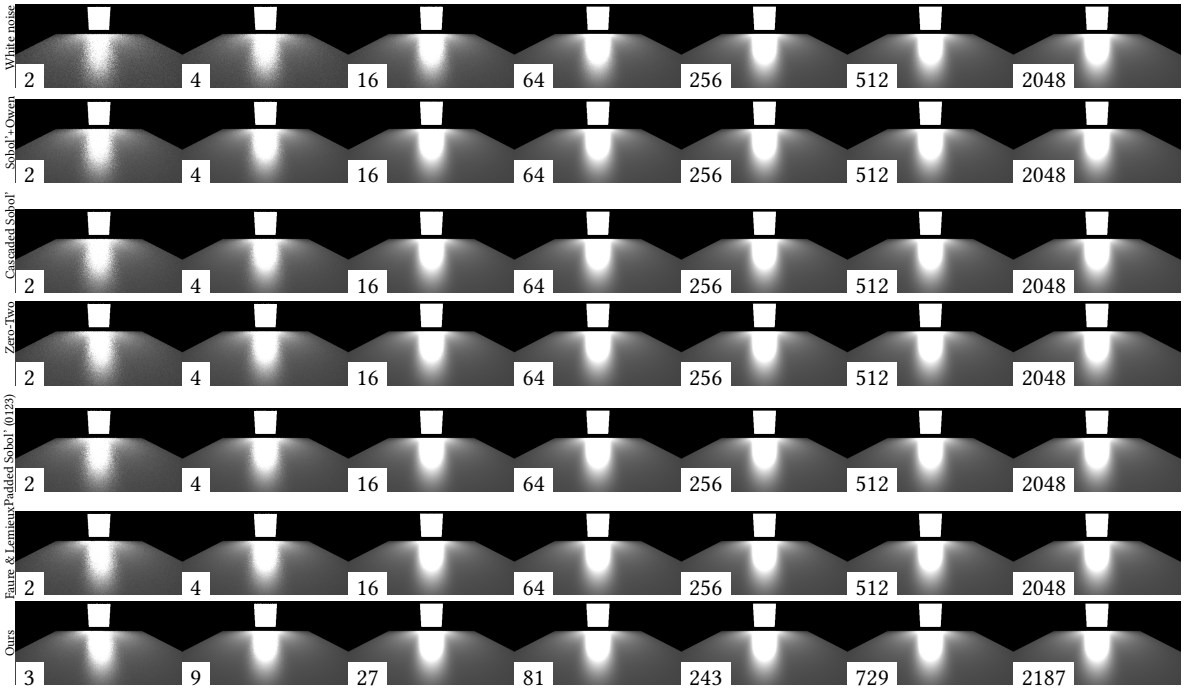

Figure 12: 6-d rendering test, 1 light source, 1 coated glossy material. Direct lighting with a multi-sample MIS estimator.

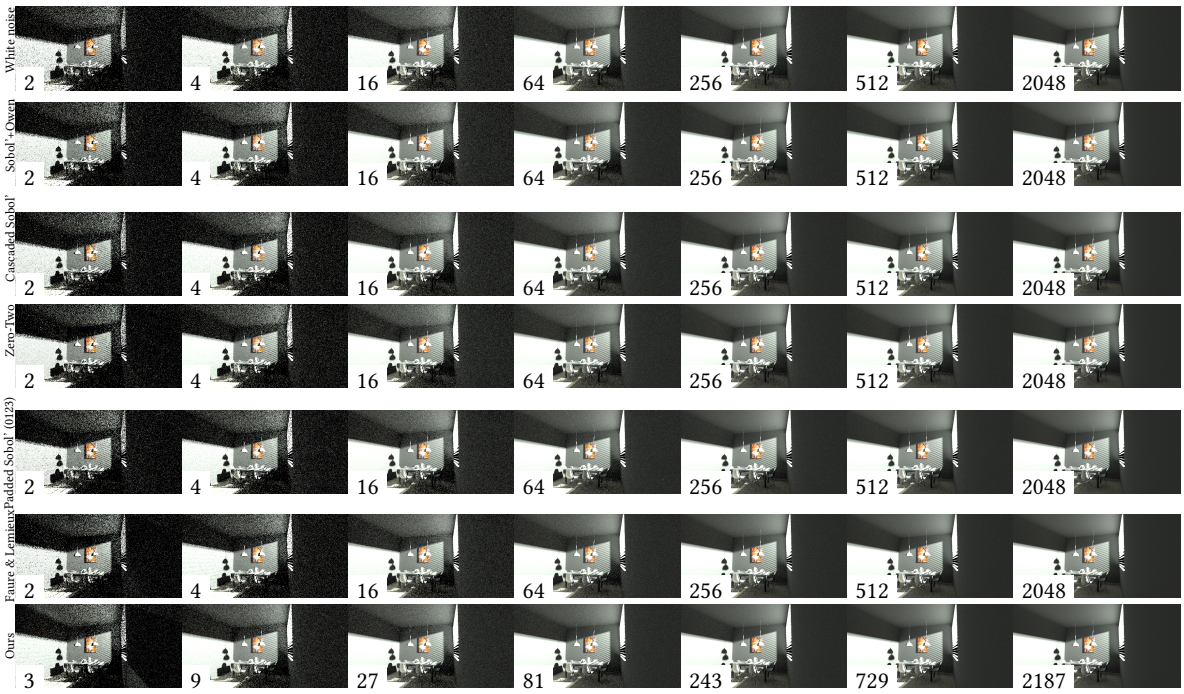

Figure 13: 10-d rendering with one bounce of indirect lighting, 1 mostly occluded light source, diffuse and glossy materials. Direct lighting with a multi-sample MIS estimator. Indirect bounce samples the material definition.

John K. Salmon, Mark A. Moraes, Ron O. Dror, and David E. Shaw. 2011. Parallel random numbers: as easy as 1, 2, 3. In *Proceedings of 2011 International Conference for High Performance Computing, Networking, Storage and Analysis (SC '11)*. Article 16, 12 pages.

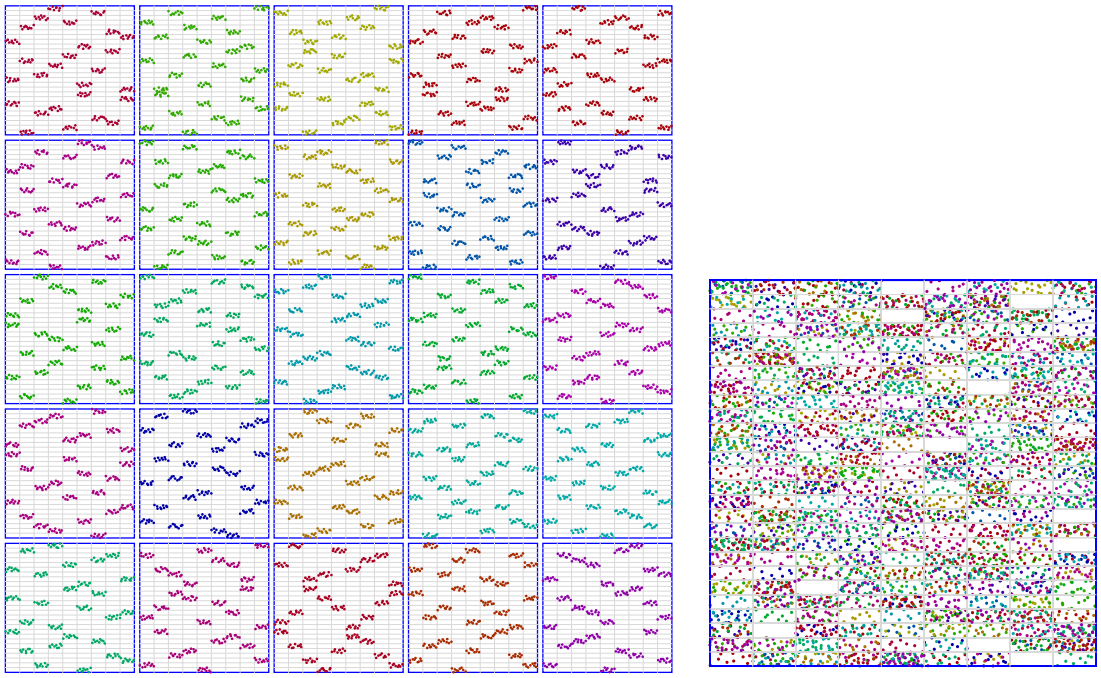

Figure 14: (Left) We illustrate the effect of Owen scrambling on an unoptimized pair of dimensions (pair $(7, 11)$) that exhibits structured blocks. We show 25 independent Owen scramblings of the same point set. Owen permutes blocks as well as points within blocks, but it does not affect the value of t (here $t = 2$, blocks are of size $1/9 \times 1/27$ and each contains either 0 or $3^t = 9$ points). (Right) However, superimposing these realizations illustrates the fact that the distribution is statistically uniform: each block on average receives the same number of points.

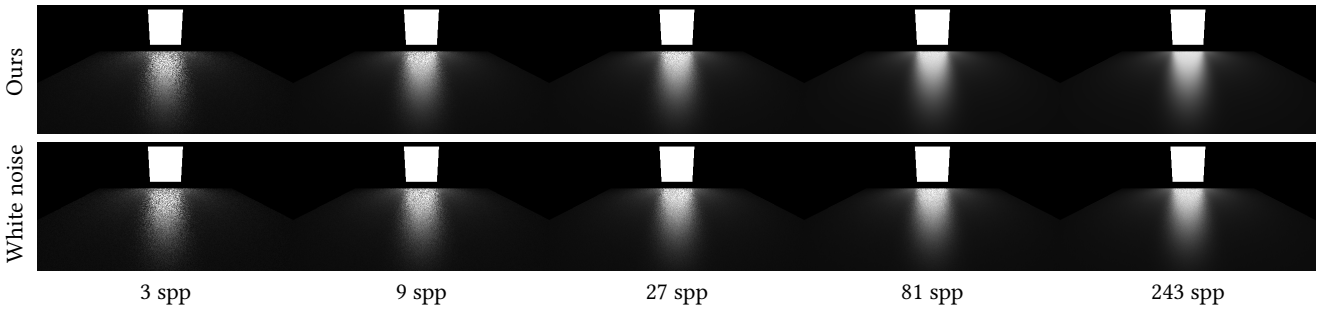

Figure 15: We illustrate rendering in 2-d with an unoptimized pair of dimensions (pair $(7, 11)$) that exhibits structured blocks, rendering with direct light only, and performing Owen scrambling for each pixel. Our rendering does not exhibit structured artifacts since Owen scrambling decorrelates blocks. Also, the lower discrepancy of our sequence, even in this case, allows to improve (top row) noise level over white noise (bottom row).