Enumerating Quasi-Monte Carlo Point Sequences in Elementary Intervals

Leonhard Grünschloß, Matthias Raab, and Alexander Keller

Abstract Low discrepancy sequences, which are based on radical inversion, expose an intrinsic stratification. New algorithms are presented to efficiently enumerate the points of the Halton and (t,s)-sequences per stratum. This allows for consistent and adaptive integro-approximation as for example in image synthesis.

1 Introduction

Similar to real world digital cameras, pixel colors can be modeled as sensor response to the radiance function. The discrete, pixel-based image thus results from projecting the radiance function onto a regular lattice of sensor functions.

These functionals can be computed by applying the Monte Carlo method on a per pixel basis, which allows one to adaptively choose the number of samples per pixel. The straightforward application of quasi-Monte Carlo methods per pixel in order to improve convergence reveals correlation artifacts, which can be removed by giving up determinism, for example by random scrambling [12, 15].

These issues can be avoided by interpreting image synthesis as a parametric integration problem, i.e. by estimating multiple functionals using a single quasi-Monte Carlo point sequence over the whole image plane: In [10] the stratification properties of the (finite) Hammersley point set have been used to efficiently map pixels to samples. This approach has been generalized for the Halton sequence in order to allow for pixel adaptive sampling [11]: As illustrated in Figure 1, a large table of

Leonhard Grünschloß
NVIDIA / Weta Digital, e-mail: leonhard@gruenschloss.org

Matthias Raab
NVIDIA, e-mail: iovis@gmx.net

Alexander Keller
NVIDIA, e-mail: keller.alexander@googlemail.com

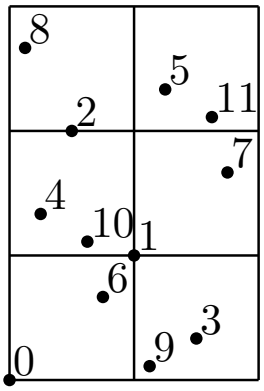

Fig. 1 A plot of the first 12 points of the scaled two-dimensional Halton sequence $(2\Phi_2(i), 3\Phi_3(i))$ labeled with their index i. While the point sequence jumps across the domain, it can be seen that points inside each of the depicted 2×3 strata can be enumerated using a stride of 6, which is the number of strata.

the size of the number of pixels has been used to look up the index of the first Halton sequence point in a pixel, while the subsequent samples have been enumerated using a fixed stride.

In the following we improve these approaches for low discrepancy sequences, whose intrinsic stratification is based on radical inversion: Algorithms, which only require a lookup table size linear in dimension, are derived for the Halton and (t,s)-sequences.

The results are applied to image synthesis, where by using the first two dimensions of the Sobol' sequence for parametric quasi-Monte Carlo integration over the whole image plane the good uniformity properties across pixels are maintained. In particular, the consistent and deterministic framework allows one to adaptively determine the number of samples per pixel according to an arbitrary density as illustrated in Figure 4.

2 Radical Inversion and Stratification

Many low discrepancy sequences are based on the principle of radical inversion

$$\begin{aligned} \Phi_b &: \mathbb{N}_0 \to \mathbb{Q} \cap [0,1) \\ i = \sum_{k=0}^{\infty} a_k(i) b^k &\mapsto \sum_{k=0}^{\infty} a_k(i) b^{-k-1}, \end{aligned} \tag{1}$$

where $a_k(i)$ denotes the $(k+1)^{\text{st}}$ digit of the integer $i \in \mathbb{N}_0$ in base b. In fact, the radical inverse (also known as van der Corput sequence [2, 13]) mirrors the digits at the decimal point. Using permutations $\sigma_b(a_k(i))$ of $\{0,\ldots,b-1\}$ instead of the

original digits can improve discrepancy [5, 10]. Note that this generalization as well as the original construction are bijections.

Inserting $i = b^d \cdot h + l$ with $l \in \{0, \ldots, b^d - 1\}$ yields

$$\Phi_b(i) = \Phi_b(b^d \cdot h + l) = b^{-d} \cdot \Phi_b(h) + \Phi_b(l), \tag{2}$$

revealing that

- the d least significant digits l select an interval $b^d \cdot \Phi_b(l) \in \{0, \ldots, b^d - 1\}$, while
- the most significant digits h determine the point inside that interval.

Therefore any subsequence of the van der Corput sequence at a step size of b^d falls into the same interval of width b^{-d}.

3 Enumerating the Halton Sequence per Stratum

The s-dimensional points

$$\mathbf{x}_i := \left(\Phi_{b_1}(i), \Phi_{b_2}(i), \ldots, \Phi_{b_s}(i)\right) \in [0,1)^s$$

constitute the Halton sequence [7], where typically b_j is the j-th prime number, although for low discrepancy it is sufficient that the b_j are relatively prime.

As illustrated in Figure 1, the stratification properties of radical inversion (2) allude to an s-dimensional stratification, where each dimension $1 \le j \le s$ is partitioned into $b_j^{d_j}$ uniform intervals for fixed $d_j \in \mathbb{N}_0$. Now, given coordinates $(p_1, \ldots, p_s)$ of such a resulting interval, where $0 \le p_j < b_j^{d_j}$, the indices

$$l_j := \Phi_{b_j}^{-1}\left(\frac{p_j}{b_j^{d_j}}\right) \in \{0, \ldots, b_j^{d_j}\}$$

uniquely identify an index $i \in \{0, \ldots, \prod_{j=1}^{s} b_j^{d_j} - 1\}$ specified by

$$l_j \equiv i \mod b_j^{d_j}, \tag{3}$$

because the bases $b_1, \ldots, b_s$ have been chosen relatively prime. Consequently the prime powers $b_j^{d_j}$ are relatively prime as well and therefore the simultaneous solution of the Equations (3) is provided by the Chinese remainder theorem [1, Sec. 31.5].

With $m_j := \left(\prod_{k=1}^{s} b_k^{d_k}\right) / b_j^{d_j}$ and the multiplicative inverse $\left(m_j^{-1} \bmod b_j^{d_j}\right)$ the index

$$i = \left(\sum_{j=1}^{s} l_j \cdot m_j \left(m_j^{-1} \bmod b_j^{d_j}\right)\right) \bmod \prod_{j=1}^{s} b_j^{d_j} \tag{4}$$

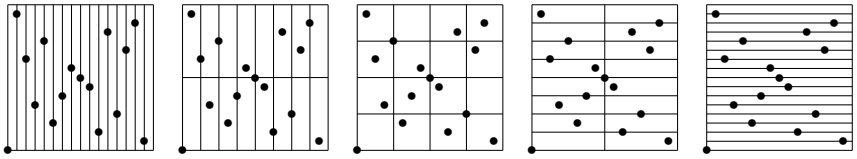

Fig. 2 All kinds of elementary intervals with area $\frac{1}{16}$ for $s = b = 2$. In this case the set of elementary intervals in the middle consists of square strata. The first $2^4 = 16$ points of Sobol's $(0,2)$-sequence, which form a $(0,4,2)$-net in base $b = 2$, are superimposed over each set of elementary intervals.

can be computed efficiently by means of the extended Euclidean algorithm [1, Sec. 31.2]. Immediate consequences are that

1. the first $\prod_{j=1}^{s} b_j^{d_j}$ points are stratified such that there is exactly one point in each stratum and that
2. all Halton sequence points with indices $i + t \cdot \prod_{j=1}^{s} b_j^{d_j}$, $t \in \mathbb{N}_0$, fall into the same stratum.

Storing a lookup table for the offsets i per stratum [11] is simple, however, the size $\prod_{j=1}^{s} b_j^{d_j}$ of the lookup table can be prohibitive even in $s = 2$ dimensions. It is much more efficient to compute the subsequence offset i by Equation (4) for a selected stratum, because only s multiplicative inverses need to be stored once.

4 Enumerating Digital (t,s)-Sequences per Elementary Interval

Opposite to Halton's construction, the components

$$x_i^{(j)} = \begin{pmatrix} b^{-1} \\ b^{-2} \\ \vdots \end{pmatrix}^T \left[C^{(j)} \begin{pmatrix} a_0(i) \\ a_1(i) \\ \vdots \end{pmatrix} \right] \in [0,1), \tag{5}$$

of digital (t,s)-sequences [13] are all generated in the same base b, while the matrix-vector multiplication takes place in a finite field. For finite fields other than $\mathbb{Z}_b$, the digits need to be mapped to the finite field and the resulting vector needs to be mapped back [13], which has been omitted for the sake of clarity. Equation (1) is an illustrative example, where the generator matrix $C^{(j)}$ is the infinite unit matrix.

The stratification properties resulting from such a construction are illustrated in Figure 2 and are formalized by

Definition 1 (see [13, p. 48]). An interval of the form

$$E(p_1,\ldots,p_s) := \prod_{j=1}^{s} \left[p_j b^{-d_j}, (p_j+1) b^{-d_j} \right) \subseteq [0,1)^s$$

for $0 \leq p_j < b^{d_j}$ and integers $d_j \geq 0$ is called an *elementary interval in base b*.

Given the numbers d_j of digits that determine the number of intervals b^{d_j} in dimension j and the elementary interval $E(p_1,\ldots,p_s)$, we have

$$\begin{pmatrix} C^{(1)}_{[(1,1),(d_1,\sum_{j=1}^{s} d_j+e)]} \\ \vdots \\ C^{(s)}_{[(1,1),(d_s,\sum_{j=1}^{s} d_j+e)]} \end{pmatrix} \cdot \begin{pmatrix} a_0(i) \\ \vdots \\ a_{\sum_{j=1}^{s} d_j-1}(i) \\ a_0(q) \\ \vdots \\ a_{e-1}(q) \end{pmatrix} = \begin{pmatrix} a_{d_1-1}(p_1) \\ \vdots \\ a_0(p_1) \\ \vdots \\ a_{d_s-1}(p_s) \\ \vdots \\ a_0(p_s) \end{pmatrix} \tag{6}$$

for the $(q+1)^{\text{st}}$ point in that elementary interval, where q constitutes the e most significant digits of the index i of that point and the shorthand

$$C^{(j)}_{[(u,v),(u',v')]} := \begin{matrix} c^{(j)}_{u,v} & c^{(j)}_{u,v+1} & \ldots & c^{(j)}_{u,v'} \\ \vdots & \vdots & \ddots & \vdots \\ c^{(j)}_{u',v} & c^{(j)}_{u',v+1} & \ldots & c^{(j)}_{u',v'} \end{matrix}$$

is used to select a block from the first d_j rows of $C^{(j)}$. As $a_0(q),\ldots,a_{e-1}(q)$ are specified by q, rearranging yields

$$\begin{aligned} &\underbrace{\begin{pmatrix} C^{(1)}_{[(1,1),(d_1,\sum_{j=1}^{s} d_j)]} \\ \vdots \\ C^{(s)}_{[(1,1),(d_s,\sum_{j=1}^{s} d_j)]} \end{pmatrix}}_{A} \cdot \begin{pmatrix} a_0(i) \\ \vdots \\ a_{\sum_{j=1}^{s} d_j-1}(i) \end{pmatrix} \\ &= \begin{pmatrix} a_{d_1-1}(p_1) \\ \vdots \\ a_0(p_1) \\ \vdots \\ a_{d_s-1}(p_s) \\ \vdots \\ a_0(p_s) \end{pmatrix} - \begin{pmatrix} C^{(1)}_{[(1,\sum_{j=1}^{s} d_j+1),(d_1,\sum_{j=1}^{s} d_j+e)]} \\ \vdots \\ C^{(s)}_{[(1,\sum_{j=1}^{s} d_j+1),(d_s,\sum_{j=1}^{s} d_j+e)]} \end{pmatrix} \cdot \begin{pmatrix} a_0(q) \\ \vdots \\ a_{e-1}(q) \end{pmatrix}, \end{aligned} \tag{7}$$

which can be solved uniquely for the index digits $a_0(i),\ldots,a_{\sum_{j=1}^{s} d_j-1}(i)$ if $\det(A) \neq 0$.

Upon existence, the inverse A^{-1} is computed once and stored for computing the indices of all samples, which in fact just costs about as much as evaluating an additional component of the sequence.

4.1 $(0,s)$-Sequences

The general definitions of (t,m,s)-nets and (t,s)-sequences in base b are based on the concept of elementary intervals (for a profound introduction see [13, Ch. 4]):

Definition 2 (see [13, Def. 4.1]). For integers $0 \leq t \leq m$, a (t,m,s)-*net in base* b is a point set of b^m points in $[0,1)^s$ such that there are exactly b^t points in each elementary interval E with volume b^{t-m}.

Definition 3 (see [13, Def. 4.2]). For an integer $t \geq 0$, a sequence $\mathbf{x}_0, \mathbf{x}_1, \ldots$ of points in $[0,1)^s$ is a (t,s)-*sequence in base* b if, for all integers $k \geq 0$ and $m > t$, the point set $\mathbf{x}_{kb^m}, \ldots, \mathbf{x}_{(k+1)b^m-1}$ is a (t,m,s)-net in base b.

According to these definitions, a $(0,s)$-sequence is a sequence of $(0,m,s)$-nets as illustrated in Figure 3. This especially includes $(0,ms,s)$-nets, where in each hypercube shaped elementary intervals of side length b^{-m}, there is exactly one point.

For the case of digital constructions, as for example the construction by Faure [4], the generator matrices $C^{(j)}$ of $(0,s)$-sequences in base b thus yield a unique solution of Equation (7). Note that $(0,s)$-sequences can only exist for $s \leq b$ [13, Cor. 4.24, p. 62].

Often integro-approximation problems expose a structure that matches uniform hypercubes like for example pixels of an image. Out of the elementary interval therefore hypercubes with $d_j = m$ are most interesting for applications. Enumerating b^e points per elementary interval thus results in $(b^m)^s \cdot b^e = b^{ms+e}$ points requiring $ms+e$ digits in total.

4.2 Sobol' Sequence

As opposed to Faure's construction, Sobol's construction [16] is restricted to base $b = 2$, which allows for $t = 0$ only up to $s = 2$ dimensions. However, the restriction to base $b = 2$ enables the use of efficient bit-vector operations [18, 6], which is not possible for $b > 2$.

The sequence can be constructed for any dimension and in fact each component is a $(0,1)$-sequence in base 2 itself. A description of how to compute the binary generator matrices can be found in [8, 9].

In addition, the first two components form a $(0,2)$-sequence in base 2 (for an efficient implementation see [12]). As a consequence the first 2^{2m} two-dimensional points are stratified such that there is exactly one point in each voxel of a $2^m \times 2^m$ regular grid over $[0,1)^2$ as illustrated in Figure 3.

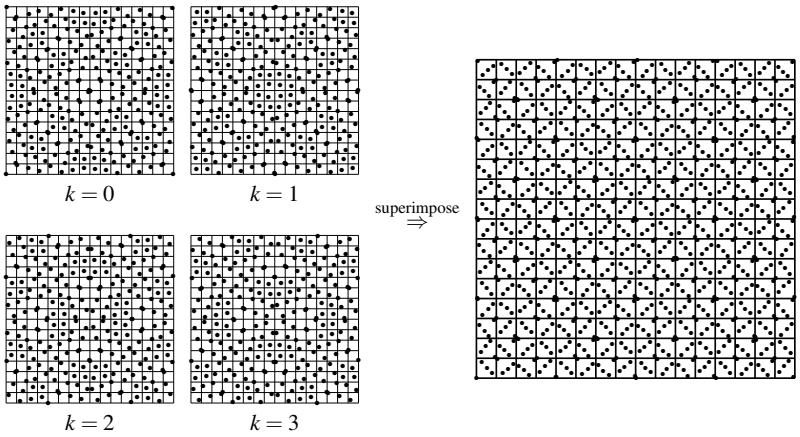

Fig. 3 Since the Sobol' sequence is a $(0,2)$-sequence, each block of points $\mathbf{x}_{k\cdot 2^8},\ldots,\mathbf{x}_{(k+1)2^8-1}$ constitutes a $(0,8,2)$-net in base 2 for $k \geq 0$. Consequently exactly one sample falls into each of the 16×16 pixels shown here. When the four consecutive sample blocks shown here are superimposed, there are four samples in each pixel. Our algorithm allows for directly enumerating these samples based on the pixel-coordinates. Note that while in this two-dimensional projection of the Sobol' sequence there are clearly some very regular patterns, the sequence is highly uniformly distributed when more dimensions are considered.

5 Consistent Image Synthesis

Partitioning the unit cube into uniform, axis-aligned intervals results in a number of strata that is exponential in the dimension s. Hence an implementation requires special attention in order to avoid overflows in standard integer arithmetic. We therefore provide illustrative source code [17] in the Python programming language, which transparently handles arbitrarily long integers.

In practice, the enumeration algorithms for both the Halton sequence and the (t,s)-sequences are useful only in small dimensions, as for example computing the average color of pixels for image synthesis. For that purpose the numbers d_j of digits are chosen such that the resulting numbers $b_j^{d_j}$ or b^{d_j}, respectively, of strata are larger or equal to the number of pixels along each dimension. While square pixels directly match the square elementary intervals of $(0,2m,2)$-nets from $(0,2)$-sequences (see Figure 3), the components of the Halton sequence need to be scaled individually per dimension [11] as illustrated in Figure 1.

Similar to [11], the entire image plane now can be sampled using only one quasi-Monte Carlo sequence, while still being able to control the sampling rate per pixel. Aside from the first two dimensions, further components are used for sampling the remaining dimensions of the integrand. This includes depth of field, area light sampling, BSDF sampling, etc. [15]. Therefore the quasi-Monte Carlo sequence needs to be extensible in the dimension like for example the Halton or Sobol' sequence.

In contrast to [11, 3] sampling per pixel is not terminated by an error threshold. Instead pixel-adaptive sampling is realized by determining a number of samples per pixel based on an error estimate, sampling each pixel according to that *speed*, and repeating this procedure, which results in a consistent integro-approximation algorithm as illustrated in Figure 4. In particular, this progressive algorithm enables strictly deterministic pixel-adaptive sampling in parallel computing environments at the cost of storing only the current number of samples per pixel.

In addition and at any point of the progressive computation a user can define pixel regions of high importance. More samples will be placed in those regions. Even with this user interaction, the determinism is not lost, i.e. if the image is accumulated up to a certain number of samples for all pixels afterwards, the user interaction does not change the result.

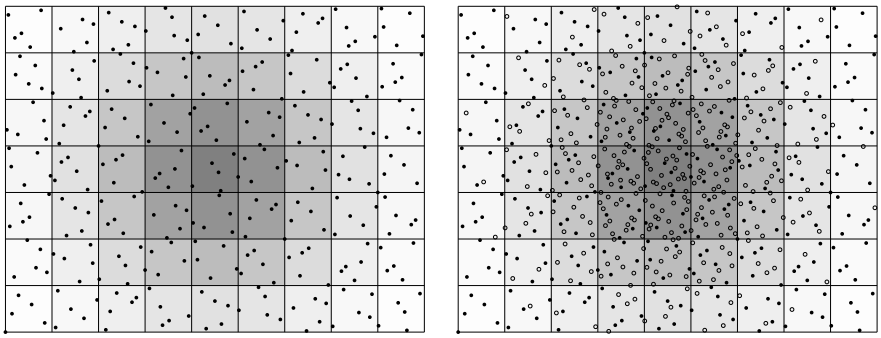

Fig. 4 On the left, 5 samples (depicted by filled circles) have been generated in each of the 9×7 pixels. Note that both the sample distributions inside each of the pixels and also across pixels is very uniform (in fact of low discrepancy). On the right, samples (depicted by stroked circles) have been added according to the underlying density proportional to the intensity level of a pixel, depicted by its gray coloring. Note that these additional samples naturally fill the remaining space. The overall distribution globally remains well distributed although additional samples have been added locally.

5.1 Enumerating the Sobol' Sequence in Pixels

The Sobol' sequence can be enumerated much more efficiently, if whole $(0, 2m, 2)$-nets (see Figure 3) are generated instead of single points. In order to explain the optimization, the index

$$i = \sum_{l=0}^{\infty} a_l(i) \cdot 2^l = \sum_{l=2m}^{\infty} a_l(i) \cdot 2^l + \underbrace{\sum_{l=m}^{2m-1} a_l(i) \cdot 2^l}_{\text{MSB}} + \underbrace{\sum_{l=0}^{m-1} a_l(i) \cdot 2^l}_{\text{LSB}}$$

is partitioned into three parts: The m least significant bits (LSB), the m most significant bits, and the remaining bits.

Now the points can be determined as follows: For each component (5) of each $(0,2m,2)$-net, two tables with 2^m entries each are computed: The first table stores the results of the matrix-vector multiplications for $0 \leq i \leq 2^m - 1$, while the second stores the results for $i = k \cdot 2^m$ for $0 \leq k \leq 2^m - 1$. A component for an arbitrary value of i then is found by looking up the entry from the first table using the LSB of i, looking up the entry from the second table using the MSB of i, and the result of the matrix-vector multiplication using the remaining bits, all combined using an exclusive-or operation. As compared to evaluating Equation (5) for each single component, the lookup tables save a lot of operations and can be considered an extension of the initial ideas in [11, Sec.2.1].

Before applying this optimization to efficiently determine the index i of each point of a $(0,2m,2)$-net of the Sobol's sequence (see Figure 3), the solution of Equation 6 for the first two components needs to be established:

Given integer pixel coordinates (p_1, p_2), with $0 \leq p_1, p_2 < 2^m$, the m least significant bits $a_0(i), \ldots, a_{m-1}(i)$ of the index i are determined by applying the inverse of $C^{(1)}$ to the bits of p_1. Then the bits of p_2 are combined with $C^{(2)}$ multiplied by the just computed least significant bits using an exclusive-or operation. Applying the inverse of $C^{(2)}$ to the result yields the most significant bits $a_m(i), \ldots, a_{2m-1}(i)$ of the index i.

By Sobol's construction, $C^{(1)}$ is a unit matrix, while $C^{(2)}$ is not, which is the reason for correcting the bits of p_2 by subtracting the contribution of the least significant bits to the most significant bits.

The optimization now consists in replacing all matrix-vector multiplications by table lookups. This requires to compute the lookup tables of size 2^m for each of the $(0,2m,2)$-nets.

The resulting implementation [17] in fact is very simple. Note that special care has to be taken in order to avoid overflows due to insufficient word width.

6 Conclusion

We derived efficient algorithms to enumerate low discrepancy sequences in elementary intervals resulting from radical inversion. These algorithms can be used for consistent deterministic parallel quasi-Monte Carlo integro-approximation.

Instead of considering all elementary intervals, it is interesting to restrict observations to $(\mathscr{M}, \mu)$-uniform point sets as introduced in [14], which includes the interesting question, whether rank-1 lattice sequences can be efficiently enumerated inside the sets of $\mathscr{M}$ [11].

References

1. Cormen, T., Leiserson, C., Rivest, R., Stein, C.: Introduction to Algorithms, Second Edition. MIT Press (2001)
2. van der Corput, J.: Verteilungsfunktionen. Proc. Ned. Akad. v. Wet. **38**, 813–821 (1935)
3. Dammertz, H., Hanika, J., Keller, A., Lensch, H.: A hierarchical automatic stopping condition for Monte Carlo global illumination. In: Proc. of the WSCG 2009, pp. 159–164 (2009)
4. Faure, H.: Discrépance de suites associées à un système de numération (en dimension s). Acta Arith. **41**(4), 337–351 (1982)
5. Faure, H.: Good permutations for extreme discrepancy. J. Number Theory **42**, 47–56 (1992)
6. Grünschloß, L.: Motion blur. Master's thesis, Ulm University (2008)
7. Halton, J.: On the efficiency of certain quasi-random sequences of points in evaluating multi-dimensional integrals. Numerische Mathematik **2**, 84–90 (1960)
8. Joe, S., Kuo, F.: Remark on algorithm 659: Implementing Sobol's quasirandom sequence generator. ACM Trans. Math. Softw. **29**(1), 49–57 (2003)
9. Joe, S., Kuo, F.: Constructing Sobol' sequences with better two-dimensional projections. SIAM Journal on Scientific Computing **30**(5), 2635–2654 (2008)
10. Keller, A.: Strictly deterministic sampling methods in computer graphics. SIGGRAPH 2003 Course Notes, Course #44: Monte Carlo Ray Tracing (2003)
11. Keller, A.: Myths of computer graphics. In: D. Talay, H. Niederreiter (eds.) Monte Carlo and Quasi-Monte Carlo Methods, pp. 217–243. Springer (2004)
12. Kollig, T., Keller, A.: Efficient multidimensional sampling. Computer Graphics Forum **21**(3), 557–563 (2002)
13. Niederreiter, H.: Random Number Generation and Quasi-Monte Carlo Methods. SIAM, Philadelphia (1992)
14. Niederreiter, H.: Error bounds for quasi-Monte Carlo integration with uniform point sets. J. Comput. Appl. Math. **150**, 283–292 (2003)
15. Pharr, M., Humphreys, G.: Physically Based Rendering: From Theory to Implementation, 2nd edition. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA (2010)
16. Sobol', I.: On the distribution of points in a cube and the approximate evaluation of integrals. Zh. vychisl. Mat. mat. Fiz. **7**(4), 784–802 (1967)
17. Downloadable source code package for this article. http://gruenschloss.org/sample-enum/sample-enum-src.zip
18. Wächter, C.: Quasi-Monte Carlo light transport simulation by efficient ray tracing. Ph.D. thesis, Ulm University (2008)