Vol. [VOL], No. [ISS]: 1–??

Golden Ratio Sequences For Low-Discrepancy Sampling

Colas Schretter and Leif Kobbelt
RWTH Aachen University, Germany

Paul-Olivier Dehaye ETH Zurich, Switzerland

Abstract. Most classical constructions of low-discrepancy point sets are based on generalizations of the one-dimensional binary van der Corput sequence whose implementation requires non-trivial bit-operations. As an alternative we introduce the quasi-regular golden ratio sequences which are based on the fractional part of successive integer multiples of the golden ratio. By leveraging results from number theory we show that point sets which evenly cover the unit square or disc can be computed by a simple incremental permutation of a generator golden ratio sequence. We compare ambient occlusion images generated with a Monte Carlo ray-tracer based on random, Hammersley, blue noise and golden ratio point sets. The source code of the ray-tracer used for our experiments is available online.

1. Low-discrepancy point sets

Hammersley A classical approach for generating quasi-random sequences of point samples in the unit square $[0,1)^2$ relies on generalizations of the van der Corput low-discrepancy sequence based on small prime numbers [Niederreiter 92]. For instance, the Hammersley point set of size N is

$$\{(H_2(i), i/N)\}_{i=1}^{N} \tag{1}$$

1086-7651/06 $0.50 per page

where $H_b(i)$ enumerates the elements of the van der Corput sequence in base b. The value $H_b(i)$ is computed by reversing the digits of the integer representation of i in base b and taking the result as the fractional part of a fixed point number. This operation can be implemented efficiently with simple bitwise operations for the special cases of bases equal to a power of two, where $b = 2^p,\ \forall p > 1$.

Halton When the number of samples is not known beforehand, Halton proposed a low-discrepancy sequence of self-avoiding points using different small prime numbers as the bases of van der Corput sequences for successive dimensions. For instance, the points

$$(H_2(i), H_3(i)), \quad \forall\, i \geq 1 \tag{2}$$

are elements of the Halton sequence in two dimensions. The rationale for this method is that indices i are expressed in base of prime numbers; therefore, the digits of representations in base 2 and 3 are less likely to be correlated. However, the discrepancy of the points $\{H_b(i)\}_{i=1}^{N}$ is far from optimal when N is not close to a power of b; hence, N needs to be close to a power of each of the different primes, which is a prohibitive restriction even in two dimensions.

Blue noise Intuitively, clustering of points should be avoided and therefore distances to the nearest neighbor point should be constant. This property is ensured only by regular grids such as an hexagonal packing of disks. Since regular sampling causes strong aliasing artifacts, it is often wanted that the point set exhibits a blue noise frequency spectrum. A blue noise pattern is a random point distribution with attenuated response in the low-frequency band. Several geometrical constructions such as Penrose tiling followed by iterative Lloyd's relaxation [Balzer et al. 09] or dart-throwing [Dunbar and Humphreys 06] have been applied for creating point sets with blue noise property. Unfortunately, those heuristic methods are rather slow and often complicated to implement.

Rank-1 lattices Alternatively, one can construct a rank-1 Fibonacci lattice for N equal to the Fibonacci number $F_k,\ \forall k \geq 3$ as follows,

$$\{(\langle i \cdot F_{k-1}/F_k \rangle, i/F_k)\}_{i=1}^{N} \tag{3}$$

where $\langle x \rangle = x - \lfloor x \rfloor$. The orthogonal projection of points on the second axis yields regular equispaced samples while the projection on the first axis shows the quasi-regular distribution generated by fractional parts of multiples of the ratios of successive Fibonacci numbers. This construction generalizes to higher dimension [Sloan and Reztsov 02]; however, this lattice rule has proven optimal discrepancy in two dimensions only [Niederreiter 92].

Unfortunately, equispacing of the coordinates in the second dimension is inherent to lattice rules based on generator vectors [Dammertz et al. 08] as well as Hammersley sets. Alignments and systematic regularities will generate aliasing artifacts in image rendering applications. In his *Instant Radiosity* paper, Keller proposed to jitter the Hammersley points as a workaround to avoid aliasing [Keller 97]. The jitter is a small random displacement applied in both dimensions since the van der Corput sequence in base 2 also generates grid-aligned coordinates.

2. Golden ratio sequences

This paper introduces the one-dimensional golden ratio sequences as an alternative to the dyadic scheme of van der Corput. We define the golden ratio sequences by the fractional parts of a seed constant $s \in [0, 1)$ plus an integer multiple of the golden ratio ϕ, i.e.,

$$G_s(i) = \langle s + i \cdot \phi \rangle, \quad \forall\, i \geq 1 \tag{4}$$

with

$$\phi = \frac{1 + \sqrt{5}}{2} \approx 1.618034\ldots \tag{5}$$

Note that the conjugate golden section $\tau = \phi - 1$ can be used in place of ϕ since only fractional parts are retained. Hence, computing $G_s(i + 1)$ given $G_s(i)$ only involves an addition and a test for integer overflow.

Golden ratio sequences are much faster and simpler to compute than van der Corput sequences, even in base 2. Furthermore, computations can be carried out most easily in the integer domain. For instance, if unsigned integers are stored in 32 bits registers, a sequence of quasi-uniform numbers in $[0, 2^{32})$ is produced by successive additions of $\lfloor \tau \cdot 2^{32} \rfloor = 2654435769$. The register can be initialized with any seed value and implicit overflows of the carry bit alleviate the need for additional comparisons.

Figure 1 compares the five first values of the van der Corput in base 2 and the golden ratio sequence with $s = 0$. In fact, the van der Corput sequence always splits the *largest* interval between points in two and the ratio of the *largest* interval to the *smallest* is two. In particular, when the number of elements is a power of the base prime, all large intervals have been broken into two small ones and the ratio is one, yielding a perfectly regular distribution. Samples from the golden ratio sequence cover more evenly the unit interval for any number of elements. In comparison to the van der Corput sequence, the distance between the two extremes points (5 and 3) is larger. Simultaneously, the largest gap between neighbor points (2 and 4) is smaller, resulting in better avoidance between samples.

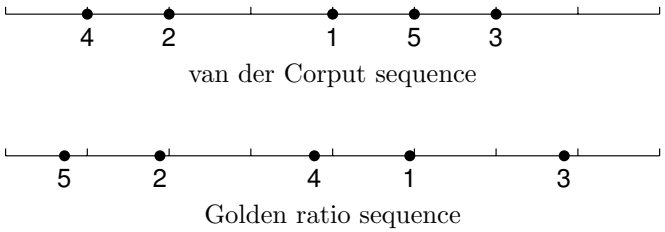


Figure 1. The five first elements of the van der Corput sequence in base 2 (top) and the golden ratio sequence with $s = 0$ (bottom).

The golden ratio conjugate τ is the most irrational[1] from all irrational numbers smaller than one [Coxeter 53]. As a consequence, the variability of pairwise distances between points is minimized when using this constant in fractional sequences. To our knowledge, the golden ratio has not yet been used as a mean for the generation of low-discrepancy sequences in computer graphics. The next section further exploits known results about this sequence and describes a very fast algorithm for generating quasi-regular point sets.

3. Golden point sets

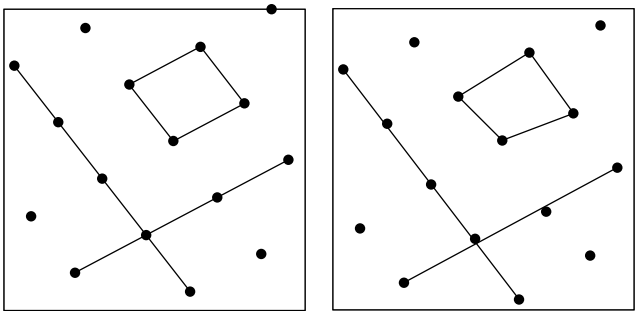
A very fast method using permutations of an initial golden ratio sequence allows the construction of low-discrepancy quasi-lattice point sets. The first coordinates of points in a golden point set are given by a golden ratio sequence, while the second coordinates are a permutation of that same sequence. The result is a irregular distribution whose orthogonal projections yields a golden ratio sequence, as shown in figure 2. This approach is inspired by the Faure point sets that are constructed from permutations of a set of coordinates that are initialized with van der Corput in base 2 [Faure 92]. Golden point sets are thus defined as

$$\{(G_s(i), G_s(\sigma_s(i)))\}_{i=1}^{N} \tag{6}$$

where the second coordinates are obtained by the permutation σ_s of $[1, \ldots, N]$ that sorts the elements of $\{G_s(i)\}_{i=1}^{N}$.

While this description involves sorting a set of values, there is an alternative algorithm that generates the same permutation without explicit sorting. It relies on the fact that ϕ can be arbitrarily well approximated by ratios of consecutive Fibonacci numbers. Therefore, as long as the denominator is greater

[1]Number theorists have related the quality of approximations of a real number by rational numbers to the size of the coefficients that appear in its continued fraction expansion. Smaller coefficients in this expansion lead to reals with worse rational approximations. The golden section ϕ, with a continued fraction expansion with constant coefficients 1 (minimal) thus has the worst approximations.



Lattice (regular) Golden set (irregular)

Figure 2. 16 points placed by a lattice rule and the similar but irregular golden point set with $s = 0$ and $N = 16$.

or equal to N, comparison of the rational approximations to the fractional parts is enough to compare the fractional parts themselves. Thereby, we can recover σ_s with a simple incremental rule as follows.

Let the increment $a = F_{2i+1}$ and the decrement $b = F_{2i}$ be the two smallest Fibonacci numbers with successive even and odd index such that $a + b \geq N$. An algorithm for mapping the elements of a golden ratio sequence towards the second dimension starts with the index $\sigma_s(1)$ initialized to the position of the minimum fractional part in the first dimension. This initial value is found by retaining the minimum when generating the first dimension.

The second coordinate of the first point is assigned with $G_s(\sigma_s(1))$. The remaining assignations are determined using a very simple iteration defined as follows:

$$\sigma_s(i+1) = \begin{cases} \sigma_s(i) + a & \text{if } \sigma_s(i) \leq b \\ \sigma_s(i) - b & \text{otherwise} \end{cases} \tag{7}$$

Note that the result can overflow after an increment by a and the assignation should simply be skipped in this case. A facultative third pass could generate another golden sequence for the first dimension, using a second seed value. This variant produces different sets of coordinates for each dimensions.

For illustration, we give an example for $N = 6$ and $s = 0$. Then, $a = F_5 = 5$, $b = F_4 = 3$ and $\sigma_0(1) = 5$. The sequence produced by the above recurrence relation is $[5, 2, \mathbf{7}, 4, 1, 6, 3]$. The boldfaced number above N should be skipped and we get the permutation $\sigma_0 = [5, 2, 4, 1, 6, 3]$. In our implementation, the permutation vector is not explicitly stored but assignments of coordinates for the second dimension are directly executed. The execution time is linear with N, as only one pass per dimension is required with a test and one addition or subtraction per element.

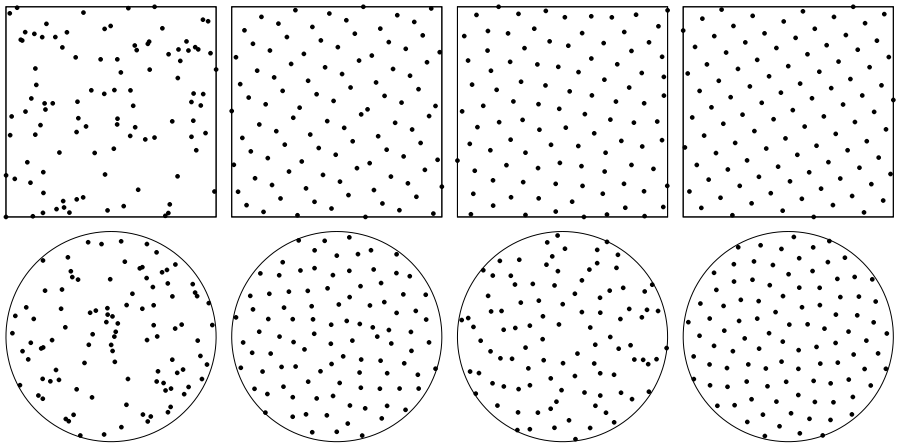


Figure 3. Comparison of 100 irregular (see Figure 2) golden points with various schemes and plots in both Cartesian (top) and polar (bottom) coordinates.

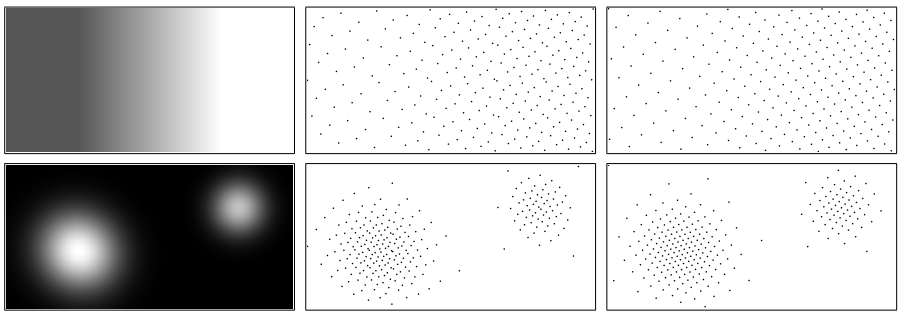
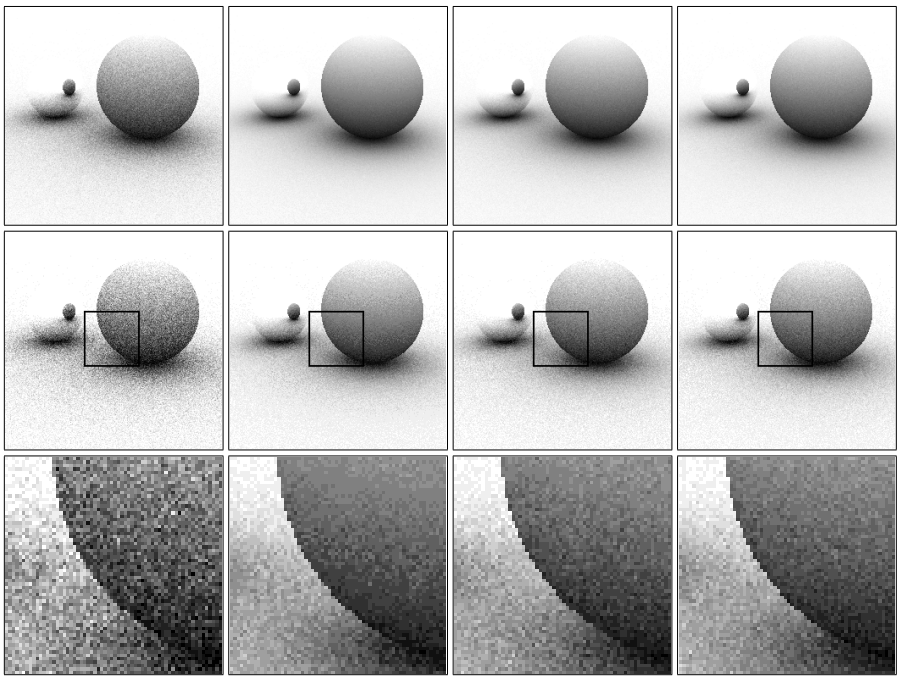


Figure 4. The Hammersley point set and a golden point set are warped according to a clamped linear ramp (top) and a mixture of two multivariate Gaussians (bottom).

4. Experiments

Figure 3 plots point sets in Cartesian and polar coordinate systems. One dimension is interpreted as a normalized radius and the second is multiplied by 2π to yield an angle. The random sequence fails to fill gaps between clusters. The other point sets exhibit the low-discrepancy property by more uniformly covering the unit square. Note that the blue noise pattern contains spurious alignments that are visible by *strings* of points in polar coordinates. In a phyllotaxis interpretation [Coxeter 53], golden point sets in cylindrical and polar coordinates are realistic models for distributing scales on the skin of pineapples and seeds in the head of sunflowers.



Random Hammersley Blue noise Golden set

Figure 5. Ambient occlusion integration with 64 (top) and 16 (middle) samples per pixel. Golden point sets yield banding-free images with low-variance.

Golden point sets inherit from the many theoretical properties of the golden ratio number. In particular, such sets are quasi-regular and thus provide natural stratification while avoiding aliasing that would arise from regular alignements. Golden point sets have been compared with the Hammersley set for adaptive sampling driven by a discrete probability density functions. Figure 4 shows warped points. A sub-pixel accurate implementation of the *inversion method* with *guide tables* for hashing [Devroye 86, Page 96] was used for warping points in amortized constant time.

Figure 5 compares ambient occlusion images computed by Monte Carlo integration. Aliasing (banding) artifacts impair the image computed with the Hammersley point set and Cranley-Patterson rotations [Kollig and Keller 02]. The blue noise point set was generated with a variant of the iterative Lloyd's algorithm [Balzer et al. 09]. The experiment demonstrates that while the blue noise model is more appropriate than Hammersley for antialiasing purpose, it performs slightly worse than our quasi-lattice rule for numerical integration. One reason is that even if blue noise point sets covers evenly the unit square, they can loosen their stratification after warping (see Fig. 3, 2^{nd} row, 3^{rd} col.).

References

[Balzer et al. 09] M. Balzer, T. Schlömer, and O. Deussen. "Capacity-Constrained Point Distributions: A Variant of Lloyd's Method." In *Proc. ACM SIGGRAPH 2009*, pp. 86–93, 2009.

[Coxeter 53] H. S. M. Coxeter. "The Golden Section, Phyllotaxis, and Wythoff's Game." *Scripta Mathematica* 19 (1953), 135–143.

[Dammertz et al. 08] S. Dammertz, H. Dammertz, and A. Keller. "Efficient Search for Low-Dimensional Rank-1 Lattices with Applications in Graphics." In *Proc. Monte Carlo and Quasi-Monte Carlo Methods*, pp. 271–287, 2008.

[Devroye 86] L. Devroye. *Non-Uniform Random Variate Generation*. Springer-Verlag, New York, 1986.

[Dunbar and Humphreys 06] D. Dunbar and G. Humphreys. "A spatial data structure for fast Poisson-disk sample generation." In *Proc. ACM SIGGRAPH 2006*, pp. 503–508, 2006.

[Faure 92] H. Faure. "Good permutations for extreme discrepancy." *Journal of Number Theory* 42 (1992), 47–56.

[Keller 97] A. Keller. "Instant Radiosity." In *Proc. ACM SIGGRAPH '97*, pp. 49–56, 1997.

[Kollig and Keller 02] T. Kollig and A. Keller. "Efficient Multidimensional Sampling." *Computer Graphics Forum* 21:3 (2002), 557–563.

[Niederreiter 92] H. Niederreiter. *Random number generation and quasi-Monte Carlo methods*, CBMS-NSF Regional Conference Series in Applied Mathematics, 63. SIAM, 1992.

[Sloan and Reztsov 02] I. H. Sloan and A. V. Reztsov. "Component-by-component construction of good lattice rules." *Mathematics of Computation* 71:237 (2002), 263–273.

Web Information:

C++ ray-tracer source code: http://jgt.akpeters.com/papers/Schretter2012/

Received [DATE]; accepted [DATE].

A. Generating golden point sets in C++

```
void golden_set(double points[][2], const unsigned int N) {
    // set the initial first coordinate
    double x = drand48();
    double min = x;
    unsigned int idx = 0;

    // set the first coordinates
    for(unsigned int i = 0; i < N; ++i) {
        points[i][1] = x;

        // keep the minimum
        if(x < min)
            min = x, idx = i;

        // increment the coordinate
        x += 0.618033988749894;
        if(x >= 1) --x;
    }

    // find the first Fibonacci >= N
    unsigned int f = 1, fp = 1, parity = 0;
    while(f + fp < N) {
        unsigned int tmp = f; f += fp, fp = tmp;
        ++parity;
    }

    // set the increment and decrement
    unsigned int inc = fp, dec = f;
    if(parity & 1)
        inc = f, dec = fp;

    // permute the first coordinates
    points[0][0] = points[idx][1];
    for(unsigned int i = 1; i < N; ++i) {
        if(idx < dec) {
            idx += inc;
            if(idx >= N) idx -= dec;
        } else
            idx -= dec;
        points[i][0] = points[idx][1];
    }
    // set the initial second coordinate
    double y = drand48();

    // set the second coordinates
    for(unsigned int i = 0; i < N; ++i) {
        points[i][1] = y;

        // increment the coordinate
        y += 0.618033988749894;
        if(y >= 1) --y;
    }
}
```