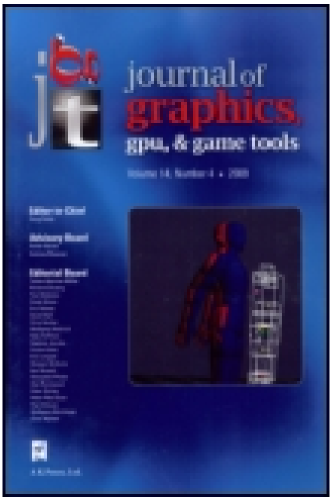


Journal of Graphics Tools
Publication details, including instructions for authors and subscription information:
http://www.tandfonline.com/loi/ujgt19

Sampling with Hammersley and Halton Points

Tien-Tsin Wong [a], Wai-Shing Luk [b] & Pheng-Ann Heng [c]

[a] Department of Computer Science and Engineering, The Chinese University of Hong Kong, Shatin, Hong Kong E-mail:

[b] Departement Computerwetenschappen, Katholieke Universiteit Leuven, Celestijnenlaan 200A, B-3001 Heverlee, Belgium E-mail:

[c] Department of Computer Science and Engineering, The Chinese University of Hong Kong, Shatin, Hong Kong E-mail:

Published online: 06 Apr 2012.

To cite this article: Tien-Tsin Wong, Wai-Shing Luk & Pheng-Ann Heng (1997) Sampling with Hammersley and Halton Points, Journal of Graphics Tools, 2:2, 9-24, DOI: 10.1080/10867651.1997.10487471

To link to this article: http://dx.doi.org/10.1080/10867651.1997.10487471

Vol. 2, No. 2: 9–24

Sampling with Hammersley and Halton Points

Tien-Tsin Wong
The Chinese University of Hong Kong

Wai-Shing Luk
Katholieke Universiteit Leuven

Pheng-Ann Heng
The Chinese University of Hong Kong

Abstract. The Hammersley and Halton point sets, two well-known, low discrepancy sequences, have been used for quasi-Monte Carlo integration in previous research. A deterministic formula generates a uniformly distributed and stochastic-looking sampling pattern at low computational cost. The Halton point set is also useful for incremental sampling. In this paper, we discuss detailed implementation issues and our experience of choosing suitable bases for the point sets, not just on the two-dimensional plane but also on a spherical surface. The sampling scheme is also applied to ray tracing, with a significant improvement in error over standard sampling techniques.

1. Introduction

Many different sampling techniques are used in computer graphics for the purpose of antialiasing. The two easiest ways to sample are randomly and regularly. Unfortunately, random sampling gives a noisy result. Regular sampling causes aliasing which requires many extra samples to reduce.

1086-7651/97 $0.50 per page

Several techniques in between random and regular sampling have been proposed. The thesis of Shirley [Shirley 91b] surveyed the common sampling techniques including jittered [Cook et al. 84], semijittered, Poisson disk, and N-rooks sampling. Cychosz generated [Cychosz 90] sampling jitters using look-up tables. Chiu et al. [Chiu et al. 94] combined jittered and N-rooks methods to design a new multijittered sampling. Cross [Cross 95] used a genetic algorithm to find the optimal sampling pattern for uniformly distributed edges. All these methods make trade-offs between noisiness and aliasing.

A sampling technique is hierarchical if, when requested to generate N_0 samples, the result coincides with the first N_0 samples it would generate in a sequence of $N = N_0 + 1$ samples. This feature is useful since the number of samples can be incrementally increased without recalculating the previous ones. Shoemake [Shoemake 91] mentioned a means to incrementally sample one-dimensional space while keeping the samples as uniform as possible. However, this method is not easily generalized to higher dimensions. Among previously mentioned methods, only Poisson disk and random sampling are hierarchical.

Discrepancy analysis measures sample point equidistribution; that is, measures how uniformly distributed the point set is. Shirley [Shirley 91a] first applied this technique to the sampling problem. The possible importance of discrepancy in computer graphics was also discussed by Niederreiter [Niederreiter 92a]. Dobkin et al. [Dobkin, Eppstein 93a], [Dobkin, Mitchell 93b], [Dobkin et al. 96] proposed various methods to measure the discrepancy of sampling patterns and to generate these patterns [Dobkin et al. 96]. Heinrich and Keller [Heinrich, Keller 94a], [Heinrich, Keller 94b], [Keller 95] and Ohbuchi and Aono [Ohbuchi, Aono 96] applied low-discrepancy sequences to Monte Carlo integration in radiosity applications.

In this paper, we discuss two useful low-discrepancy sequences, namely Hammersley and Halton. These sequences have been used in numerical [Paskov, Traub 95], [Traub 96], [Case 95] and graphical [Heinrich, Keller 94a], [Heinrich, Keller 94b], [Keller 95], [Ohbuchi, Aono 96] applications, with a significant improvement in terms of error. Previous research mainly concentrated on sample generation on the two-dimensional plane, cube, and hypercube. Recently researchers have found [Cui, Freeden 97] that mapping Hammersley points with base of two to the surface of a sphere also gives uniformly distributed directional vectors. We discuss the implementation issues and experience in choosing suitable bases of Hammersley and Halton points on the two-dimensional plane and spherical surface.

The mathematical formulation is briefly described in Section 2. Section 3 compares sampling patterns generated using different bases. Ray-tracing experiments to verify the usefulness of the method are discussed in Section 4 The Appendix lists C implementations.

2. Hammersley and Halton Points

We first describe the definition of Hammersley and Halton points and then discuss their implementation in detail. For more mathematical specifics, readers are referred to the more mathematically based literature [Niederreiter 92b], [Cui, Freeden 97].

Each nonnegative integer k can be expanded using a prime base p:

$$k = a_0 + a_1 p + a_2 p^2 + \ldots + a_r p^r, \tag{1}$$

where each a_i is an integer in $[0, p-1]$. Now we define a function Φ_p of k by

$$\Phi_p(k) = \frac{a_0}{p} + \frac{a_1}{p^2} + \frac{a_2}{p^3} + \cdots + \frac{a_r}{p^{r+1}}. \tag{2}$$

The sequence of $\Phi_p(k)$, for $k = 0, 1, 2, \ldots$, is called the van der Corput sequence [Tezuka 95].

Let d be the dimension of the space to be sampled. Any sequence $p_1, p_2, \ldots, p_{d-1}$ of prime numbers defines a sequence $\Phi_{p_1}, \Phi_{p_2}, \ldots, \Phi_{p_{d-1}}$ of functions whose corresponding k-th d-dimensional Hammersley point is

$$\left(\frac{k}{n}, \Phi_{p_1}(k), \Phi_{p_2}(k), \ldots, \Phi_{p_{d-1}}(k)\right) \quad \text{for } k = 0, 1, 2, \ldots, n-1. \tag{3}$$

Here $p_1 < p_2 < \cdots < p_{d-1}$ and n is the total number of Hammersley points. To evaluate the function $\Phi_p(k)$, the following algorithm can be used.

$$\begin{array}{l} p' = p \;,\; k' = k \;,\; \Phi = 0 \\ \text{while } k' > 0 \text{ do} \\ \qquad a = k' \bmod p \\ \qquad \Phi = \Phi + \frac{a}{p'} \\ \qquad k' = \texttt{int}(\frac{k'}{p}) \\ \qquad p' = p'p \end{array}$$

where $\texttt{int}(x)$ returns the integer part of x.

The above algorithm has a complexity of $O(\log_p k)$ for evaluating the k-th point. Hence the worst-case bound of the algorithm for generating $(N+1)$ points is

$$\begin{aligned} & \log_p(1) + \log_p(2) + \cdots + \log_p(N-1) + \log_p(N) \\ \le\; & \log_p(N) + \log_p(N) + \cdots + \log_p(N) + \log_p(N) \\ =\; & N \log_p N. \end{aligned}$$

A Pascal implementation of this algorithm can be found in [Halton, Smith 64]. In most computer graphics applications, the dimension of the sampled space is either two or three. In this paper, we concentrate on the generation of a uniformly distributed point set on the surface of the two-dimensional plane and sphere using Hammersley points. Higher dimensional sets can be similarly generated using Formulas (1)–(3).

2.1. Points on the Two-Dimensional Plane

On the two-dimensional plane, Formula (3) simplifies to

$$\left(\frac{k}{n}, \Phi_{p_1}(k)\right) \quad \text{for } k = 0, 1, 2, \ldots, n-1. \tag{4}$$

The range of $\frac{k}{n}$ is $[0, 1)$, while that of $\Phi_{p_1}(k)$ is $[0, 1]$. For computer applications, a good choice of the prime p_1 is $p_1 = 2$. The evaluation of $\Phi_2(k)$ can be done efficiently with about $\log_2(k)$ bitwise shifts, multiplications, and additions: no division is necessary. The C implementation of two-dimensional Hammersley points with base two is shown in the Appendix (Source Code 5.). We shift $\frac{k}{n}$ by 0.5 to center the sequence. Otherwise, $\Phi_{p_1}(0)$ will always equal zero for any n, which is a undesirable effect.

However, the original Hammersley algorithm is not hierarchical, due to the first coordinate $\frac{k}{n}$, which for different values of n results in different sets of points. This problem can be resolved by using two p-adic van der Corput sequences with different prime numbers p_1 and p_2. This hierarchical version is known as the Halton point set [Niederreiter 92b], [Tezuka 95].

$$(\Phi_{p_1}(k), \Phi_{p_2}(k)) \quad \text{for } k = 0, 1, 2, \ldots, n-1. \tag{5}$$

Since both functions $\Phi_{p_1}(k)$ and $\Phi_{p_2}(k)$ are hierarchical (being independent of n by construction), the Halton point sets are hierarchical as well. Source Code 5. in the Appendix implements the Halton point sets on the two-dimensional plane.

2.2. Points on the Sphere

To generate directional vectors, or (equivalently) points on the spherical surface, the following mappings [Spanier, Gelbard 69] are needed:

$$\left(\frac{k}{n}, \Phi_p(k)\right) \mapsto (\phi, t) \mapsto \left(\sqrt{1-t^2}\cos\phi, \sqrt{1-t^2}\sin\phi, t\right)^T . \tag{6}$$

The first, from $\left(\frac{k}{n}, \Phi_p(k)\right)$ to (ϕ, t), is simply a linear scaling to the required cylindrical domain, $(\phi, t) \in [0, 2\pi) \times [-1, 1]$. The mapping from (ϕ, t) to $(\sqrt{1-t^2}\cos\phi, \sqrt{1-t^2}\sin\phi, t)^T$ is a z-preserving radial projection from the unit cylinder $C = \{\,(x, y, z) \mid x^2 + y^2 = 1 \;\; |z| \le 1\}$ to the unit sphere.

As before, the coordinate $\frac{k}{n}$ makes the scheme nonhierarchical. Halton points on the sphere can be generated in a similar manner by using two p-adic van der Corput sequences with different prime bases.

$$(\Phi_{p_1}(k), \Phi_{p_2}(k)) \mapsto (\phi, t). \tag{7}$$

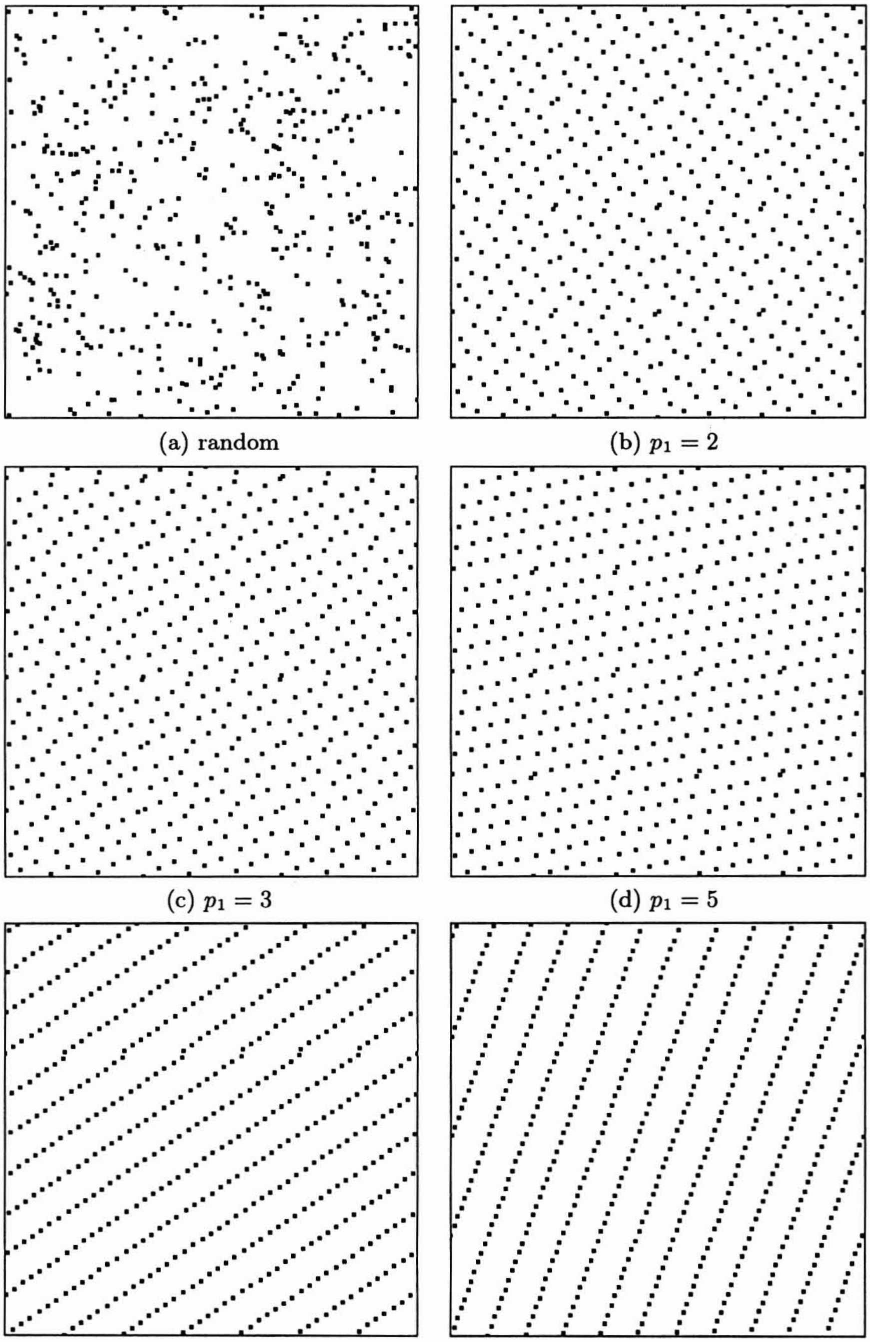
Source Code 5. in Appendix A shows the C implementation of Hammersley points on the sphere, with a similar 0.5 shift applied to prevent a fixed sample point from appearing at the south pole. Source Code 5. shows the Halton point version. For efficiency of computation, we fixed $p_1 = 2$ while leaving p_2 as a user input. This restriction can be trivially removed.

3. Appearance

Figures 1 and 2 show the Hammersley points with different bases, on the plane and sphere respectively. We generated 500 samples for the planar test and 1000 for the spherical test. Figures 1(a) and 2(a) are the patterns of random sampling on the plane and sphere respectively. Compared to the random sampling pattern (Figure 1(a)), the Hammersley point set with $p_1 = 2$ (Figure 1(b)) gives a pleasant, less clumped pattern. The points are uniformly distributed without a perceptible pattern. Among the patterns with different bases, a Hammersley point set with $p_1 = 2$ (Figure 1(b)) also gives the most uniformly distributed pattern. As the base p_1 increases (from Figures 1(b)–1(f)), the pattern becomes more and more regular. The points tend to line up in slanting lines, which will clearly increase aliasing problems.

The same progression affects spherical sampling patterns (Figures 2(b)–2(f)). When $p_1 = 2$, it gives the best uniformly distributed pattern on the sphere. Cui et al. [Cui, Freeden 97] measured the uniformity of Hammersley points with $p_1 = 2$ on the sphere using the generalized discrepancy. Hammersley points with $p_1 = 2$ give the lowest generalized discrepancy (most uniformly distributed) among the methods tested. As p_1 increases (from Figures 2(b)–2(f)), points start to line up and form regular lines on the sphere. The position of the pole (marked with an arrow) becomes distinguishable from the pattern.

The Halton point sets give patterns with varying uniformity and regularity (Figures 3 and 4). To compare the effect of different bases p_1 and p_2, all patterns generated with $p_1 = 2$ are placed on the left, and those with $p_1 = 3$ on the right. The omission of the case where $p_1 = p_2 = 3$ is due to the constraint $p_1 < p_2$. Figure 3(b) gives a pattern with somewhat aligned points. Others give rather pleasant appearances. Among the point sets tested, none gives a better pattern than Hammersley points with $p_1 = 2$. In general, the patterns of Halton points are quite unpredictable. Nevertheless, after transforming the points to the sphere, the pole and equator of the sphere become indistinguishable (Figures 4(a)–4(e)). They are not as uniformly distributed as the Hammersley point set with $p_1 = 2$, but no lining-up occurs like that observed in Hammersley points.

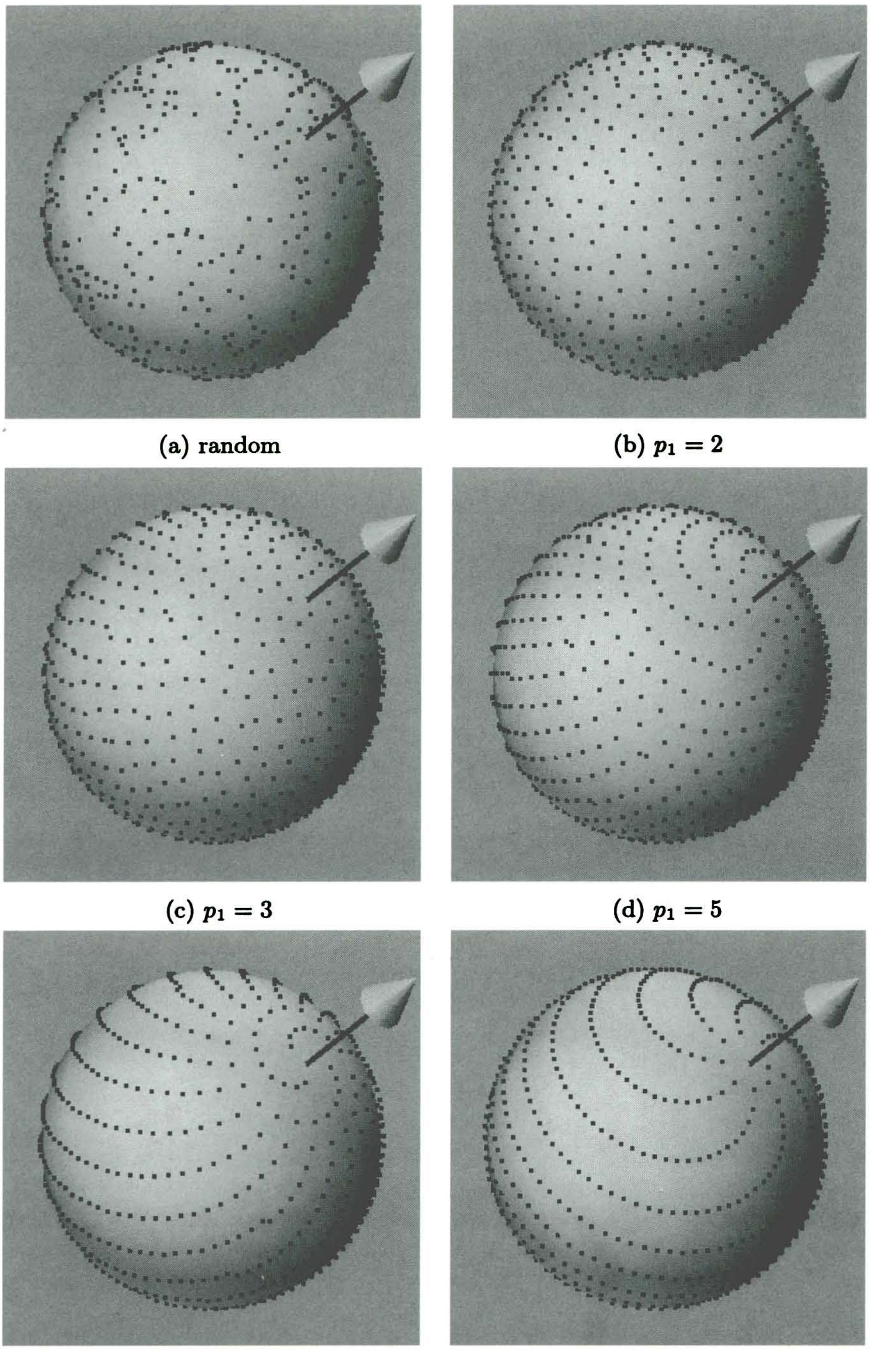


(a) random (b) $p_1 = 2$

(c) $p_1 = 3$ (d) $p_1 = 5$

(e) $p_1 = 7$ (f) $p_1 = 11$

Figure 1. Hammersley points on the two-dimensional plane ($n = 500$).



(a) random

(b) $p_1 = 2$

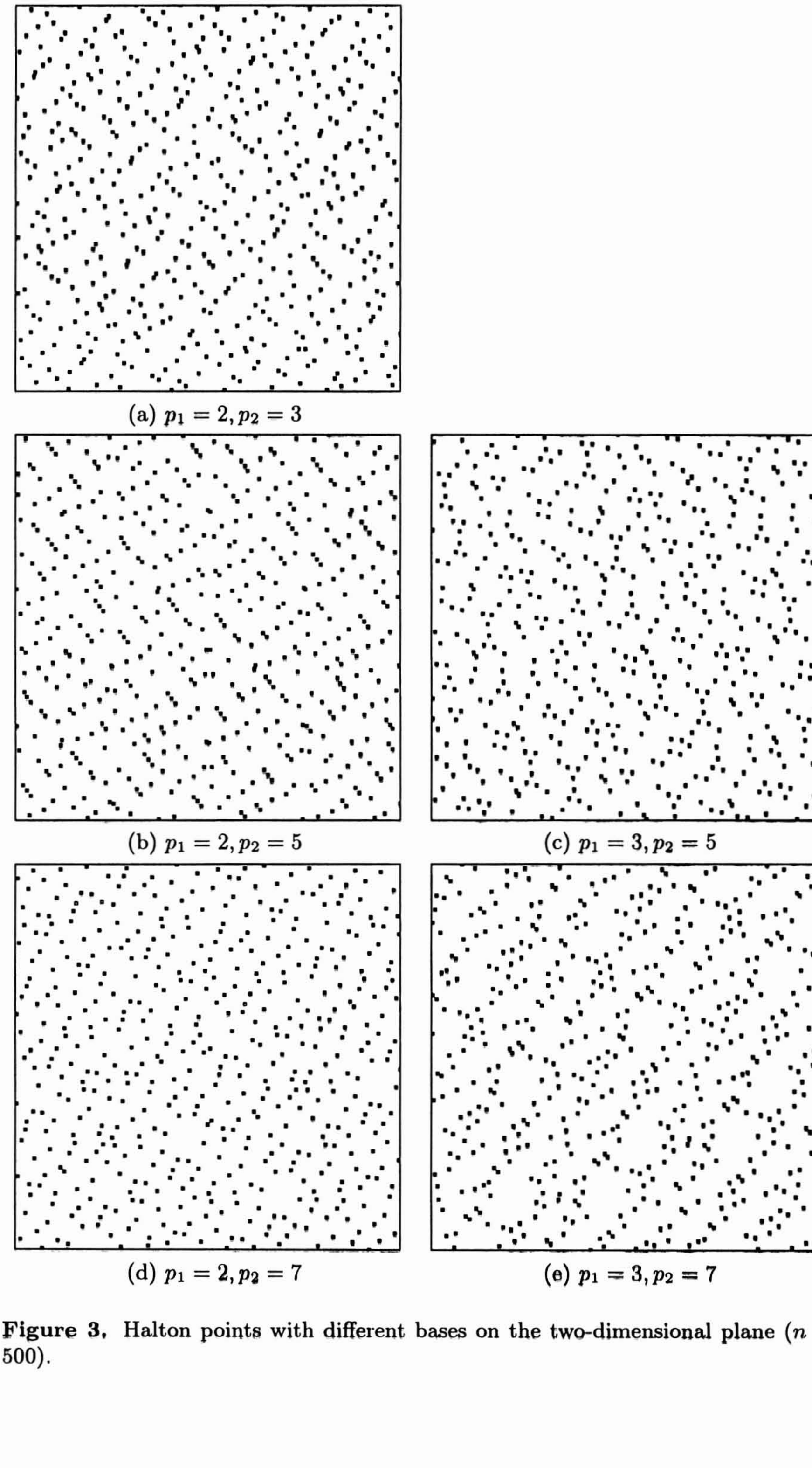
(c) $p_1 = 3$

(d) $p_1 = 5$

(e) $p_1 = 7$

(f) $p_1 = 11$

Figure 2. Hammersley points on the sphere ($n = 1000$).



(a) $p_1 = 2, p_2 = 3$

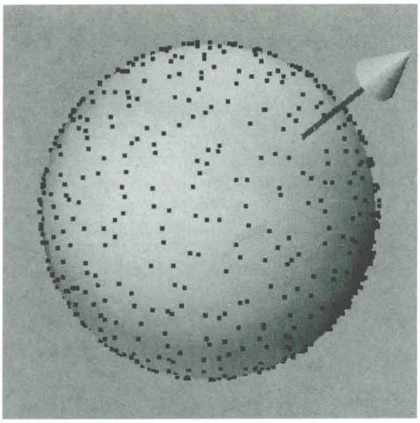
(b) $p_1 = 2, p_2 = 5$

(c) $p_1 = 3, p_2 = 5$

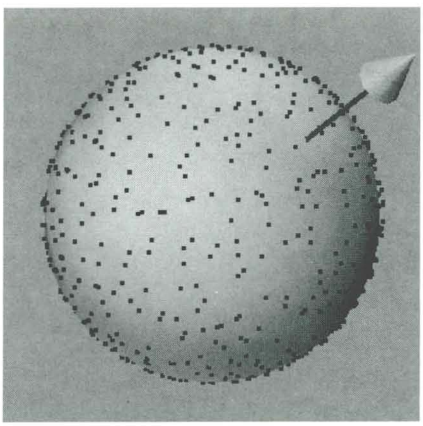
(d) $p_1 = 2, p_2 = 7$

(e) $p_1 = 3, p_2 = 7$

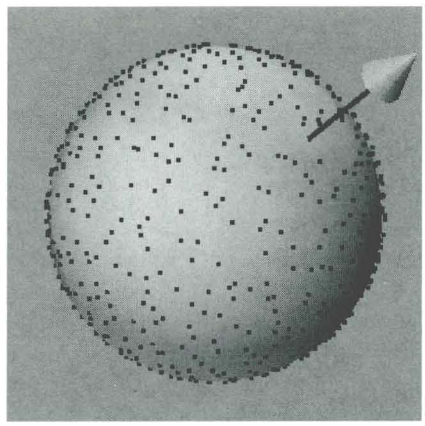
Figure 3. Halton points with different bases on the two-dimensional plane ($n =$ 500).



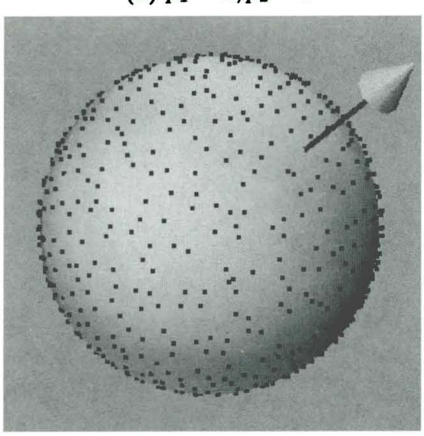
(a) $p_1 = 2, p_2 = 3$



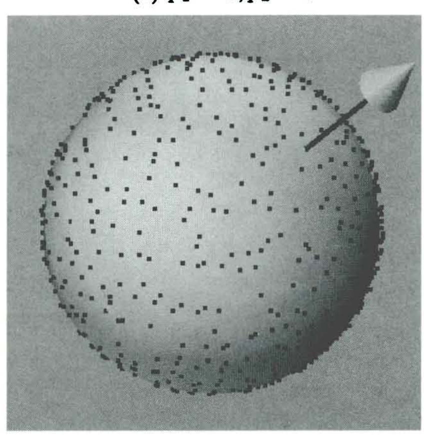
(b) $p_1 = 2, p_2 = 5$



(c) $p_1 = 3, p_2 = 5$

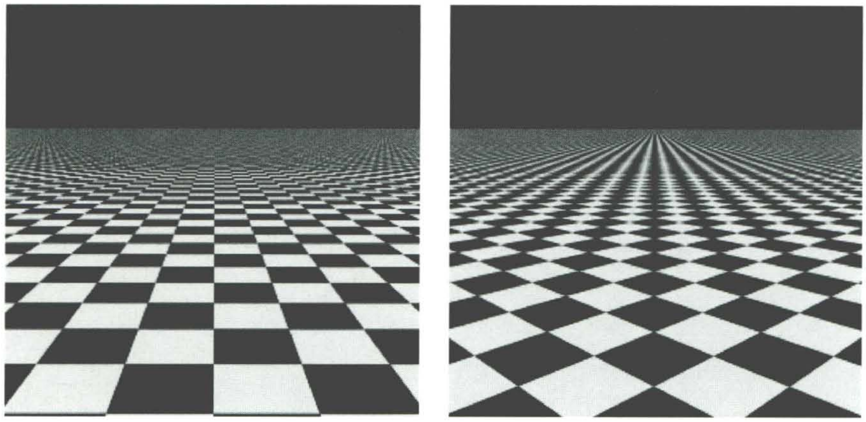


(d) $p_1 = 2, p_2 = 7$



(e) $p_1 = 3, p_2 = 7$

Figure 4. Halton points with different bases on the sphere ($n = 1000$).



(a)**checker** (b)**checker45**

Figure 5. The two test scenes used in the sampling test.

4. Ray-Tracing Experiments

The method is tested in a ray tracer. Instead of generating a distinct sampling pattern for each pixel, a single sampling pattern is generated for the whole screen. Otherwise, the sampling pattern for each pixel would be the same, since the Hammersley and Halton points are actually deterministic. Hence, we can only specify the *average* sample per pixel.

Two scenes are chosen for testing: **checker** (Figure 5(a)) and **checker45** (Figure 5(b)). The "correct" images, used for calculating the pixel error E in luminance, are produced by sampling the scenes using jittered sampling with 400 samples per pixel. Five other sampling schemes—jittered, multi-jittered, Poisson disk, random, and regular—are included for comparison. All of these five sampling schemes are tested with 16 samples per pixel, while the Hammersley and Halton point sets are tested with an average of 16 samples per pixel.

Four statistical data are recorded: average (Mean$|E|$), standard deviation (S.D.($|E|$)), root-mean-square (R.M.S.(E)), and maximum (Max.($|E|$)) of the absolute pixel error in luminance. Tables 1 and 2 show the statistics from test scenes **checker** and **checker45** respectively. Methods listed in the tables are ranked by their performance.

Among the tested methods, the Hammersley point set with $p_1 = 2$ gives the lowest average standard derivation and root-mean-square of absolute pixel errors in both test scenes. Multijittered sampling is the first runner-up. Hammersley point sets with higher bases ($p_1 > 3$) are not tested due to the lining-up phenomenon, which certainly introduces aliasing. For Halton point sets, we arbitrarily choose two bases for testing since there is no general trend in

Methods	Mean($\|E\|$)	S.D.($\|E\|$)	R.M.S.(E)	Max.($\|E\|$)
Hamm., $p_1 = 2$	0.0086	0.0247	0.0261	0.3451
multi-jitter, $n = 4, N = 16$	0.0091	0.0261	0.0277	0.3843
Hamm., $p_1 = 3$	0.0097	0.0265	0.0282	0.3961
Halton, $p_1 = 2, p_2 = 7$	0.0105	0.0280	0.0299	0.3451
Halton, $p_1 = 2, p_2 = 3$	0.0110	0.0291	0.0312	0.3686
jittered, 4×4	0.0128	0.0335	0.0358	0.3804
Poisson, $d = 0.2$	0.0132	0.0338	0.0363	0.3804
random	0.0179	0.0443	0.0478	0.3961
regular	0.0188	0.0491	0.0526	0.5098

Table 1. Statistics of the ray-traced image `checker`. E is the pixel error in luminance.

Methods	Mean($\|E\|$)	S.D.($\|E\|$)	R.M.S.(E)	Max.($\|E\|$)
Hamm., $p_1 = 2$	0.0101	0.0264	0.0282	0.3882
multi-jitter, $n = 4, N = 16$	0.0103	0.0270	0.0289	0.3686
Hamm., $p_1 = 3$	0.0106	0.0274	0.0294	0.4431
Halton, $p_1 = 2, p_2 = 3$	0.0114	0.0287	0.0309	0.3882
Halton, $p_1 = 2, p_2 = 7$	0.0131	0.0289	0.0310	0.4118
jittered, 4×4	0.0131	0.0332	0.0357	0.3765
Poisson, $d = 0.2$	0.0133	0.0332	0.0358	0.4118
regular	0.0138	0.0393	0.0416	0.5059
random	0.0185	0.0446	0.0483	0.4000

Table 2. Statistics of the ray-traced image `checker45`. E is the pixel error in luminance.

the appearance of the patterns. In our experiment, Hammersley point sets are better than the tested Halton point sets. Both Hammersley and Halton point sets give lower error than that of traditional jittered and Poisson disk sampling except the multijittered method.

5. Conclusion

The Hammersley point set with $p_1 = 2$ gives the most uniformly distributed sampling pattern. For higher p_1, the points tend to align and reduce its usefulness. Although the Halton point sets do not give patterns as uniformly distributed as Hammersley point sets, they do not have the line-up problem and allow incremental sampling.

Hammersley points and Halton points have proved useful for quasi-Monte Carlo integration. The methods have been applied to ray-tracing applications with a significant improvement in pixel error. The complexity of both the Hammersley and Halton points' generation algorithms is $O(N \log_p N)$, which is smaller than that of Poisson disk.

Acknowledgments. We would like to thank Prof. Timothy Poston of National University of Singapore for his careful proofread and useful suggestions. We would also like to express our thanks to the editor and reviewers for their valuable advice and comments.

Appendix: Source Code

Source Code 1. *Hammersley Points on Two-Dimensional Plane with* $p_1 = 2$

```
void PlaneHammersley(float *result, int n)
{
  float p, u, v;
  int k, kk, pos;

  for (k=0, pos=0 ; k<n ; k++)
  {
    u = 0;
    for (p=0.5, kk=k ; kk ; p*=0.5, kk>>=1)
      if (kk & 1)                              // kk mod 2 == 1
        u += p;
    v = (k + 0.5) / n;
    result[pos++] = u;
    result[pos++] = v;
  }
}
```

Source Code 2. *Halton Points on Two-Dimensional Plane with* $p_1 = 2$

```
void PlaneHalton(float *result, int n, int p2)
{
  float p, u, v, ip;
  int k, kk, pos, a;

  for (k=0, pos=0 ; k<n ; k++)
  {
    u = 0;
    for (p=0.5, kk=k ; kk ; p*=0.5, kk>>=1)
      if (kk & 1)                             // kk mod 2 == 1
        u += p;
    v = 0;
    ip = 1.0/p2;                              // inverse of p2
    for (p=ip, kk=k ; kk ; p*=ip, kk/=p2)  // kk = (int)(kk/p2)
      if ((a = kk % p2))
        v += a * p;
    result[pos++] = u;
    result[pos++] = v;
  }
}
```

Source Code 3. *Hammersley Points on Sphere with* $p_1 = 2$

```
void SphereHammersley(float *result, int n)
{
  float p, t, st, phi, phirad;
  int k, kk, pos;

  for (k=0, pos=0 ; k<n ; k++)
  {
    t = 0;
    for (p=0.5, kk=k ; kk ; p*=0.5, kk>>=1)
      if (kk & 1)                             // kk mod 2 == 1
        t += p;
    t = 2.0 * t  - 1.0;                       // map from [0,1] to [-1,1]
    phi = (k + 0.5) / n;                      // a slight shift
    phirad =  phi * 2.0 * M_PI;               // map to [0, 2 pi)
    st = sqrt(1.0-t*t);
    result[pos++] = st * cos(phirad);
    result[pos++] = st * sin(phirad);
    result[pos++] = t;
  }
}
```

Source Code 4 *Halton Points on Sphere with* $p_1 = 2$

```
void SphereHalton(float *result, int n, int p2)
{
  float p, t, st, phi, phirad, ip;
  int k, kk, pos, a;

  for (k=0, pos=0 ; k<n ; k++)
  {
    t = 0;
    for (p=0.5, kk=k ; kk ; p*=0.5, kk>>=1)
      if (kk & 1)                            // kk mod 2 == 1
        t += p;
    t = 2.0 * t - 1.0;                       // map from [0,1] to [-1,1]
    st = sqrt(1.0-t*t);
    phi = 0;
    ip = 1.0/p2;                             // inverse of p2
    for (p=ip, kk=k ; kk ; p*=ip, kk/=p2)    // kk = (int)(kk/p2)
      if ((a = kk % p2))
        phi += a * p;
    phirad =  phi * 4.0 * M_PI;              // map from [0,0.5] to [0, 2 pi)
    result[pos++] = st * cos(phirad);
    result[pos++] = st * sin(phirad);
    result[pos++] = t;
  }
}
```

References

[Case 95] James Case. "Wall Street's Dalliance with Number Theory." *SIAM News* (December 1995).

[Chiu et al. 94] Kenneth Chiu, Peter Shirley, and Changyaw Wang. "Multi-Jittered Sampling." In *Graphics Gems IV*, edited by Paul S. Heckbert, p. 370–374. Cambridge, MA: AP Professional, 1994.

[Cook et al. 84] Robert L. Cook, Thomas Porter, and Loren Carpenter. "Distributed Ray-Tracing." *Computer Graphics (Proc. SIGGRAPH 84)*, 18(3):137–145(July 1984).

[Cross 95] Robert A. Cross. "Sampling Patterns Optimized for Uniform Distribution of Edges." In *Graphics Gems V*, edited by Alan W. Paeth, p. 359–363. Chestnut Hill, MA: AP Professional, 1995.

[Cui, Freeden 97] Jianjun Cui and Willi Freeden. "Equidistribution on the Sphere." *SIAM Journal on Scientific Computing*, 18(2):595–609(March 1997).

[Cychosz 90] Joseph M. Cychosz. "Efficient Generation of Sampling Jitter Using Look-Up Tables." In *Graphics Gems*, edited by Andrew Glassner, p. 64–74. Cambridge, MA: AP Professional, 1990.

[Dobkin, Eppstein 93a] D. P. Dobkin and D. Eppstein. "Computing the Discrepancy." In *Proceedings of the 9th ACM Symposium on Computational Geometry*, p. 47–52. New York: ACM, 1993.

[Dobkin et al. 96] D. P. Dobkin, David Eppstein, and Don P. Mitchell. "Computing the Discrepancy with Applications to Supersampling Patterns." *ACM Transactions on Graphics*, 15(4):354–376(October 1996).

[Dobkin, Mitchell 93b] D. P. Dobkin and D. P. Mitchell. "Random-Edge Discrepancy of Supersampling Patterns." In *Graphics Interface*, p. 62–69. Toronto: Canadian Information Processing Society, 1993.

[Halton, Smith 64] J. H. Halton and G. B. Smith. "Radical-Inverse Quasi-Random Point Sequence." *Communications of the ACM*, 7(12):701–702(December 1964).

[Heinrich, Keller 94a] Stefan Heinrich and Alexander Keller. "Quasi-Monte Carlo Methods in Computer Graphics, Part I: The qmc Buffer." Technical Report 242/94, University of Kaiserlautern, 1994.

[Heinrich, Keller 94b] Stefan Heinrich and Alexander Keller. "Quasi-Monte Carlo Methods in Computer Graphics, Part II: The Radiance Equation." Technical Report 243/94, University of Kaiserlautern, 1994.

[Keller 95] Alexander Keller. "A Quasi-Monte Carlo Algorithm for the Global Illumination Problem in the Radiosity Setting." In *Proceedings of Monte Carlo and Quasi-Monte Carlo Methods in Scientific Computing*, p. 239–251. New York: Springer-Verlag, 1995.

[Niederreiter 92a] H. Niederreiter. "Quasirandom Sampling Computer Graphics." In *Proceedings of the 3rd International Seminar on Digital Image Processing in Medicine*, p. 29–33. Riga: Latvian Academy of Sciences, 1992.

[Niederreiter 92b] H. Niederreiter. *Random Number generation and Quasi-Monte Carlo Methods*. Philadephia: CBMS-NSF and SIAM, 1992.

[Ohbuchi, Aono 96] Ryutarou Ohbuchi and Masaki Aono. "Quasi-Monte Carlo Rendering with Adaptive Sampling." Technical Report, Tokyo Research Laboratory, IBM Japan Ltd., 1996.

[Paskov, Traub 95] S. H. Paskov and J. F. Traub. "Faster Valuing of Financial Derivatives." *Journal of Portfolio Management*, 22:113–120(1995).

[Shirley 91a] Peter Shirley. "Discrepancy as a Quality Measure for Sample Distributions." In *Proceedings of Eurographics*, edited by Werner Purgathofer, p. 183–193. Amsterdam: North-Holland, 1991.

[Shirley 91b] Peter Shirley. "Physically Based Lighting Calculations for Computer Graphics." PhD Thesis, University of Illinois at Urbana-Champaign, 1991.

[Shoemake 91] Ken Shoemake. "Interval Sampling." In *Graphics Gems II*, edited by James Arvo, p. 394–395. San Diego, CA: AP Professional, 1991.

[Spanier, Gelbard 69] Jerome Spanier and Ely M. Gelbard. "Monte Carlo Principles and Neutron Transport Problems." New York, Addison-Wesley, 1969.

[Tezuka 95] Shu Tezuka. "Uniform Random Numbers: Theory and Practice." Boston: Kluwer Academic Publishers, 1995.

[Traub 96] Joseph Traub. "In Math We Trust." *What's Happening in the Mathematical Sciences*, 3:101–111(1996).

Web Information:

All source codes in the Appendix and a demonstration program showing the appearances of various Hammersley and Halton point sets are available at http://www.acm.org/jgt/papers/WongLukHeng97.

Tien-Tsin Wong, Department of Computer Science and Engineering,
The Chinese University of Hong Kong, Shatin, Hong Kong (ttwong@acm.org)

Wai-Shing Luk, Departement Computerwetenschappen, Katholieke Universiteit Leuven, Celestijnenlaan 200A, B-3001 Heverlee, Belgium
(Wai-Shing.Luk@cs.kuleuven.ac.be)

Pheng-Ann Heng, Department of Computer Science and Engineering,
The Chinese University of Hong Kong, Shatin, Hong Kong (pheng@cse.cuhk.edu.hk)

Received February 6, 1997; accepted April 23, 1997