Volume 0 (1981), Number 0 pp. 1–6

A Comparison of Tabular PDF Inversion Methods

David Cline Anshuman Razdan Peter Wonka

Arizona State University

Abstract

The most common form of tabular inversion used in computer graphics is to compute the cumulative distribution table of a pdf and then search within it to transform points, using an $O(\log n)$ binary search. Besides the standard inversion method, however, several other discrete inversion algorithms exist that can perform the same transformation in $O(1)$ time per point. In this paper, we examine the performance of three of these alternate methods, two of which are new.

Categories and Subject Descriptors (according to ACM CCS): I.3.0 [Computer Graphics]: General

1. Introduction

A number of applications in computer graphics require the generation of sample points that are distributed according to some specified distribution. For example, importance sampling reduces the variance of a Monte Carlo integral estimate by distributing sample points as closely as possible according to the function being integrated. Stippling applications also require points that are distribued according to a specified density to guide the placement of stipples. In fact, a number of object placement algorithms rely on point sets that are both well distributed and conforming to a particular density (e.g. [DMS06, KCODL06]).

The classic manner of achieving a specific sample density is to start with uniformly distributed points and then transform them to the desired density by some form of numerical inversion. If the underlying probability distribution (pdf) is defined in closed form, it may be possible to symbolically invert the corresponding cumulative distribution function (cdf). This is commonly done, for example, with analytical BRDFs.

On the other hand, if the desired distribution is defined by a discrete probability table, the usual inversion method is to compute the cdf of the table and then perform an $O(\log n)$ binary search within the cdf to transform each point, where n is the size of the table. Some work has been done to compress the cdf table while retaining the important features [LRR05], but in this work we are primarily concerned with methods that speed up point transformation rather than with methods that save space.

There are a number of problem contexts in which many points must be transformed very quickly. For example, bidirectional and resampled importance sampling [BGH05, TCE05] require the generation of a large number of tentative samples for each ray cast in a Monte Carlo rendering setting. In these techniques, the vast majority of the tentative samples are discarded, making quick sample generation paramount. Compounding this issue, ray casting itself has become cheaper with the development of packet and frustum-based ray tracing techniques [RSH05, WSBW01] so that sample generation is now becoming as much of a perfomance bottleneck as ray casting itself. Another motivation for simplifying the point transformation routine is to make it more amenable to GPU implementation. If the transformation code will be mapped to graphics hardware, it may be beneficial to use a simpler routine that does not waste valuable shader ops on loops and other branching structures. Based on these observations, it makes sense to revisit the topic of sample generation from tabular pdfs.

1.1. Overview

This paper compares a number of alternate tabular inversion algorithms to the standard binary search method for transforming points based on a tabular pdf. The algorithms that we will compare against the standard method are the *alias method*, *integer cdf inversion* followed by linear search, and

© The Eurographics Association and Blackwell Publishing 2008. Published by Blackwell Publishing, 9600 Garsington Road, Oxford OX4 2DQ, UK and 350 Main Street, Malden, MA 02148, USA.

approximate tabular cdf inversion. To our knowledge, the latter two of these methods have not been presented previously. By our comparisons we will show that each of the alternative inversion methods, and in particular the latter two, possess significant advantages over the binary search method, both in terms of speed and code parsimony, while sharing similar startup and memory costs.

2. Existing Inversion Methods for Tabular PDFs

2.1. The Binary Search Method (Standard Method)

A tabular probability distribution function is an array of values that defines a probability function. For example, figure 1 shows a tabular pdf with 8 elements.

table index	0	1	2	3	4	5	6	7
PDF: $f=$	$\frac{1}{32}$	$\frac{1}{16}$	$\frac{1}{4}$	$\frac{1}{16}$	$\frac{1}{8}$	$\frac{5}{32}$	$\frac{7}{32}$	$\frac{3}{32}$

Figure 1: *Tabular probability distribution function.*

The pdf table may represent a discrete or continuous probability distribution in one or more dimensions. When representing a continuous distribution, the pdf is usually considered to be piecewise constant, with each table entry defining the probability for a small area of the function domain.

Given a 1D pdf table, f, the corrseponding cumulative distribution function, F, is defined by the equation $F_i = \sum_{j=0}^{i} f_j$. Using the observation that $F_i = F_{i-1} + f_i$, the cdf can be initialized from f in linear time by simply looping over the values of F, computing each one in turn. Figure 2 gives the cdf table corresponding to the pdf in figure 1.

	0	1	2	3	4	5	6	7
CDF: $F=$	$\frac{1}{32}$	$\frac{3}{32}$	$\frac{11}{32}$	$\frac{13}{32}$	$\frac{17}{32}$	$\frac{11}{16}$	$\frac{29}{32}$	1

Figure 2: *Tabular cumulative distribution function.*

In 2D, a cdf table can be constructed by making a separate cdf for each row and an additional cdf for the y marginal (the table of the row sums). Once F has been computed, it can be used to transform uniform points in $[0,1)^2$ to the desired distribution by performing two binary searches within F, one for each dimension. Each of these searches takes $O(\log n)$ steps to complete.

2.2. The Alias Method

An alternative to binary searching within the cdf table is to use the *alias method* [Vos91, Wal77]. Introduced to the graphics community by Burke [Bur04], the alias method works by setting up an *alias table* whereby sample locations with excess probability can point to locations with insufficient probability. Each entry in the alias table consists of an alias location along with a probability of reassigning the point to that location.

While slightly more complex to initialize than a cdf, an alias table can be initialized in linear time as well. The algorithm to create the table starts by partitioning the elements of the pdf into two groups, those with insufficient probability, and those with excess probability. An element from each group is then chosen, and the element with excess probability P assigns its excess to the element with insufficient probability Q. In other words, P is set to point to Q. P is then removed from the list, and Q is checked to see if it now has too much probability. If Q does have too much, it is moved to the "excess probability" list. Otherwise it is left in the "insufficient probability" list. This process repeats until both lists are empty. Burke [Bur04] describes this process in detail, and provides sample code for the initialization process. Figure 3 shows an alias table created for the example pdf in figure 1. An alias table can be built for a 2D pdf by ignoring the fact that it is 2D, allowing table entries to point to each-other without restriction.

	0	1	2	3	4	5	6	7
alias index:	2	5	2	6	6	2	5	6
alias prob:	3/4	1/2	0	1/2	0	1/4	0	1/4

Figure 3: *Alias table.*

To transform points from a uniform distribution to the desired distribution, the sampling routine looks up the location of the input point in the alias table and then either returns that point unmodified, with probability $1-p$, or returns the aliased location, with probability p, where p is the probability stored in the alias table. The 2D coordinate of a transformed point can be calculated based on the array index of the aliased location.

Since no searching is involved, transforming points with the alias method is quite fast compared to the standard method. The transformation code only requires a single table lookup, and one if statement. Despite its speed, however, the alias method has a critical flaw that makes it unsuitable for many applications–it tends to disrupt the stratification or disrepancy-reduction efforts that went into creating the input point set. This effect can be seen in figures 8 and 9. The explanation for the loss of good sample spacing can be found in the manner in which the alias method redistributes samples. Instead of warping sample space to take on the desired distribution, the alias method steals points from low density regions and stuffs them into high density regions randomly, resulting in poor stratification. Another, less critical issue with the alias method is that the alias table cannot be directly queried to determine f for a given point in the domain. Consequently, applications may need to retain an extra copy of the probability table.

3. Methods Based On Approximate CDF Inversion

The reason that applications do not simply invert the cdf table is that it cannot be done exactly given the constraints

© The Eurographics Association and Blackwell Publishing 2008.

of the table. Nevertheless, approximate inversions are possible. In this section we describe two new approximate cdf inversion methods that allow points to be transformed to a desired distribution in $O(1)$ time per point, providing substantial time savings over the binary search method.

3.1. Integer CDF Inversion with Linear Search

The most expensive part of the standard method for searching within a cdf is the binary search, which must be performed for each sample point. One way to optimize the search would be to create some kind of oracle to tell the search routine where to start. If the oracle can provide initial guesses that are sufficiently close to the correct value, we can replace the binary search with a simpler linear search that runs in $O(1)$ time on average. In fact, a discrete version of the inverse cdf, $\hat{F}^{-1}$, meets the criteria just described (see appendix A for a proof). The elements of $\hat{F}^{-1}$ are defined as follows:

$$\hat{F}_i^{-1} = j : F_{j-1} \leq \frac{i}{w} < F_j. \tag{1}$$

That is, the i^{th} element of $\hat{F}^{-1}$ contains the smallest index j such that F_j is greater than i/w, where w is the size of $\hat{F}^{-1}$ (8 in our example): Note that since the values of $\hat{F}^{-1}$ are monotonically non-decreasing, the table can be initialized in linear time. Figure 4 shows $\hat{F}_i^{-1}$ for the example pdf in figure 1.

$\hat{F}^{-1} =$

0	1	2	3	4	5	6	7
0	2	2	3	4	5	6	6

Figure 4: *Integer-based inverse cdf table.*

As is the case with the cdf, $\hat{F}^{-1}$ can be initialized in 2D by inverting each of the table rows and the y axis marginal separately. Transforming a point (x,y) to (x',y') using $\hat{F}^{-1}$ table can be performed as follows:

1. Using y, look up (in the y marginal of $\hat{F}^{-1}$) the starting point for searching for y' in F.
2. Search in the marginal of F to find y'.
3. Using y', determine the row in which to search for x'.
4. Using x, look up the starting point for searching for x'.
5. Search in F to determine the value of x'.

Note that the above algorithm is essentially a form of hashing in which the inverted cdf table acts as the hash function, and the cdf table resolves hash collisions. The benefit of integer cdf inversion is that it is faster than the standard method while producing exactly the same result. Thus, integer cdf inversion tends to preserve sample spacing better than the alias method. The code is somewhat simpler than the standard method as well, but a loop construct is still needed to perform the search. While the memory requirements are about double the standard method, this is still likely to be less than the memory used by the pdf itself if it contains spectral values (BRDFs and environment maps, for example).

3.2. Approximate CDF Inversion

In the methods presented so far, the tabular pdf has been treated as an absolute ideal when in fact most of the time it is a sampled approximation. For many applications, it may be acceptable to use an approximate inverted cdf table. Code for the resulting inversion algorithm would not require any control structures, and would be able to transform points with a single table lookup in 1D, and two lookups in 2D.

The approximate inverted cdf table. We define an approximate inverted cdf table, F^{-1}, that stores real-valued coordinates instead of integer table indices. To start with, let $F(x)$ be the linearly interpolated value of F at point x. That is, given that $j = \lfloor x \rfloor$,

$$F(x) = F_{j-1} + (x-j)(F_j - F_{j-1}).$$

F_i^{-1} is defined as the coordinate x such that $F(x)$ equals i/w, where once again w is the table width:

$$F_i^{-1} = x : F(x) = \frac{i}{w}. \tag{2}$$

The value of F_i^{-1} can be calculated by interpolating based on values in F:

$$F_i^{-1} = j + (\frac{i}{w} - F_{j-1})/(F_j - F_{j-1}), \tag{3}$$

where $j = \hat{F}_i^{-1}$. Figure 5 shows the approximate inverse cdf table created from the pdf in figure 1.

$F^{-1} =$

0	1	2	3	4	5	6	7
0	$2\frac{1}{8}$	$2\frac{5}{8}$	$3\frac{1}{2}$	$4\frac{3}{4}$	$5\frac{3}{5}$	$6\frac{2}{7}$	$6\frac{6}{7}$

Figure 5: *Approximate inverse cdf table.*

As with several of the other methods discussed so far, F^{-1} can be created for a 2D pdf by inverting the table rows and y axis marginal separately.

Figure 6 shows the kinds of approximation error that can occur because of the tabular encoding of F^{-1}. The probability function in the figure represents a fairly difficult case because it has sharp features. Even in this case, however, the overall error quickly diminishes as the table size increases, and the 64^2 table is almost indistinguishable from the original.

In 2D, an input point (x,y) can be transformed to an output point, (x',y') based on two table lookups as follows:

1. Look up y' in the marginal of F^{-1} using y.
2. Using y', calculate the row in which to look up x'.
3. Look up x' in the specified row of F^{-1} using x.

© The Eurographics Association and Blackwell Publishing 2008.

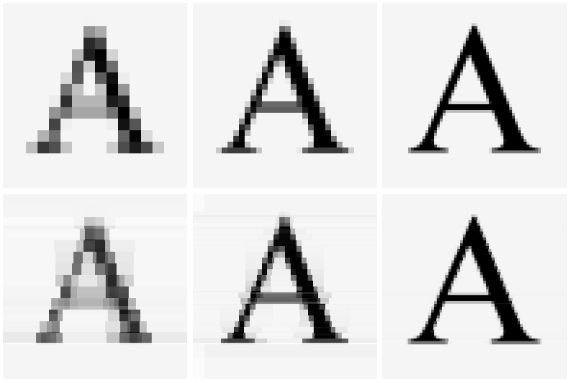


Figure 6: *Approximation error of F^{-1} for different table sizes. The top row shows tabular pdfs with resolution 16^2, 32^2 and 64^2. The bottom row shows the approximations achieved by approximate inversion with the same table size.*

The probability of the output point, $p(x',y')$, can also be calculated during the transformation:

$$p(x',y') = 1 \,/\, (M_{a+1} - M_a)(R_{c,b+1} - R_{c,b}), \qquad (4)$$

where $a = \lfloor yh \rfloor$, $b = \lfloor xw \rfloor$, $c = \lfloor y'h \rfloor$, M is the marginal of F^{-1}, R_c is row c of F^{-1}, and w and h are the width and height of F^{-1}.

The main strength of using an approximate inverted cdf table is that points can be transformed very quickly using a few table lookups. The transformation code itself is also quite simple, requiring no loop structures or "if" statements. Furthermore, point sets transformed using F^{-1} tend to maintain their good spacing properties as well as the cdf searching algorithms. The F^{-1} table also has the same memory overhead as the cdf table. The main drawback of using approximate cdf inversion is that it does not exactly reproduce the probability table. One consequence of the approximation is that the probability of an output point cannot easily be calculated without having its corresponding input point as well. Thus, the algorithm may be unsuitable for multiple importance sampling.

4. Comparison

Memory consumption. Table 1 gives the memory usage for the different inversion methods described in the paper, assuming that both integers and real values are 4 bytes in size. The alternate methods all use one or two times as much memory as the standard method, so memory consumption is not likely to be more of a concern than it is with the standard method. However, the numbers given only apply to inversion tables with the same number of entries. If we want to represent a continuous probability function, the approximate cdf inversion method may be more accurate than the other methods since it concentrates effort in high probability regions. This is similar to the environment map compression technique described by Lawrence et al. [LRR05].

Binary search method	4 bytes
The alias method	8 bytes
Integer cdf inversion	8 bytes
Approximate cdf inversion	4 bytes

Table 1: *Memory consumption per table entry for different tabular inversion methods.*

Timing. Figure 7 shows timing results for each of the four inversion methods described in the paper. As demonstrated in the figure, the performance of the standard binary search method is strongly dependent on the pdf table size. On the other hand, the other three methods are only slightly dependent on table size due to their constant time complexity. The standard method is fairly fast, transforming between 4.7 and 11.8 million points per second. However, even this rate may be a bottleneck for some applications. For example, MLRTA [RSH05] has been shown to achieve ray casting rates in the tens of millions per second. This fact underscores the need for extremely fast inversion routines to meet the demands of high performance ray tracers. All of the alternate methods achieve sampling rates that are between two and three and half times faster than the standard method, but once again, the alias method tends to de-stratify an input sequence, leaving the approximate cdf inversion techniques as the candidates of choice among the alternates.

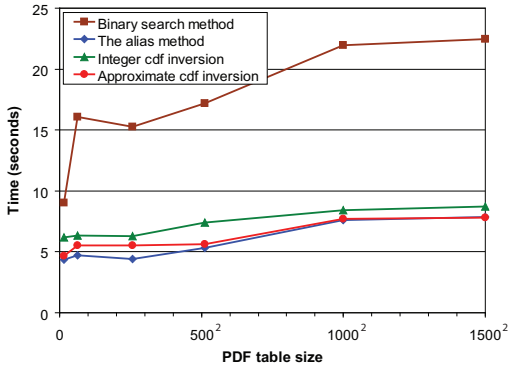


Figure 7: *Time to transform 107 million points for different pdf table sizes and inversion methods.*

Point set quality. Figure 8 shows point sets generated by the four methods described in the paper, using the same Poisson-disk input sequence. One drawback with all the cdf-based methods is that samples may squeeze together when the sample domain is warped anisotropically. This is still preferable to the randomization that occurs in the alias method for most applications, however. Direct methods [KCODL06, Ost07] can generate non-uniform point sets

© The Eurographics Association and Blackwell Publishing 2008.

of higher quality, but will likely be an order of magnitude slower.

A standard tool for judging the quality of a 2D point set is to view its Fourier transform. For a uniform point set, the Fourier transform should be radially symmetric and have a low energy ring around the origin, called the "blue noise" property. [LD06] surveys algorithms to generate blue noise point sets. Figure 9 shows point sets generated by the methods described in the paper along with their Fourier transforms. Note that the Fourier transforms of both the cdf search methods and approximate cdf inversion have a low energy anulus around the origin (excluding a cross of low frequencies). This reflects the fact that the good spacing qualities of the input point set are retained by these methods. On the other hand, the Fourier transform of the alias method has much more energy near the origin. In many ways, this Fourier transform looks like a blend between that of the input point set and a random point set. This should not be surprising, given the sample clumping that occurs in the spatial domain.

In figures 10 and 11, we show the effect of the different inversion routines on rendering convergence. Figure 10 provides renderings of a scene lit by an environment map, using the different inversion methods to choose sampling directions. With 16 samples per pixel, the cdf search methods do fairly well. As one might expect, the disruption of the sample spacing caused by the alias method results in a noisier image. Approximate inversion results in a few bright samples, but the overall quality is better than with the alias method. These results are mirrored in figure 11, which plots the RMSE for the methods in the paper using different numbers of samples.

5. Conclusion

In this paper we compared three alternate methods for transforming points according to a tabular pdf to the standard method based on binary search within the cdf table. The alternate methods included the alias method, integer cdf inversion with linear search, and approximate cdf inversion. We demonstrated that the new methods can transform points several times faster than the binary search method, while having comparable memory overhead. Based on our results we believe the alternate methods, and particularly the two based on approximate cdf inversion, to be well suited to a number of sampling applications. A possible direction for future work would be to speed up the point transformation process by processing multiple points at once to take advantage of cache coherence.

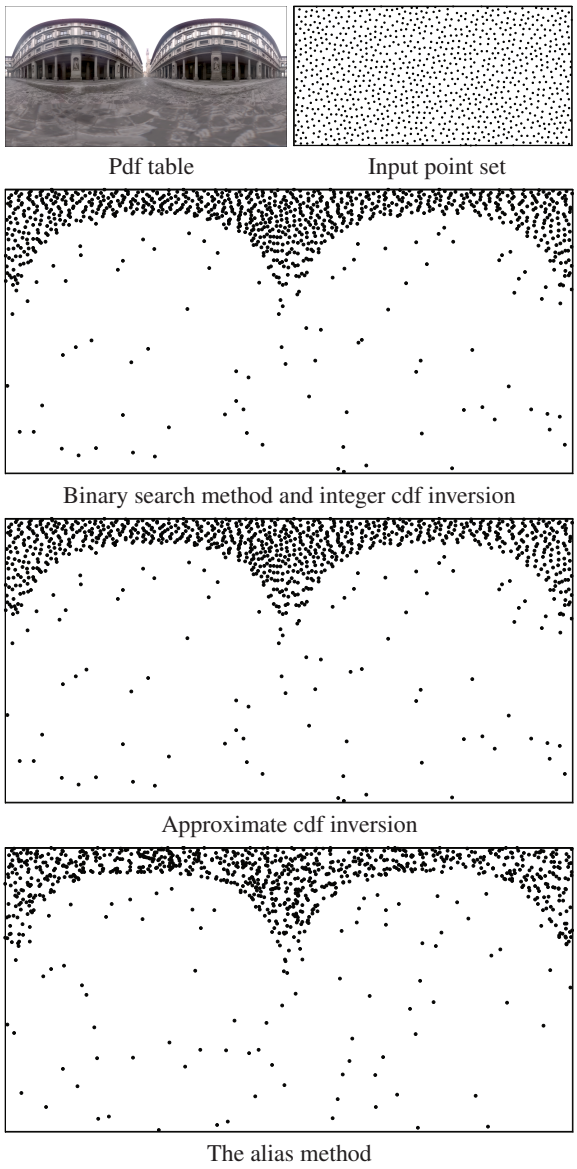


Figure 8: *Transformed point sets for different tabular inversion methods.*

References

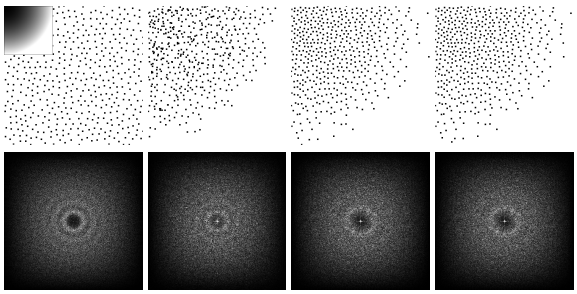
[BGH05] BURKE D., GHOSH A., HEIDRICH W.: Bidirectional importance sampling for direct illumination. In *Eurographics Symposium on Rendering* (2005), Eurographics Association, pp. 139–146.

[Bur04] BURKE D.: Bidirectional importance sampling for illumination from environment maps. *Master's Thesis, University of British Columbia* (2004).

[DMS06] DIETRICH A., MARMITT G., SLUSALLEK P.: Terrain guided multi-level instancing of highly complex plant populations. In *Proceedings of the 2006 IEEE Symposium on Interactive Ray Tracing* (September 2006), pp. 169–176.

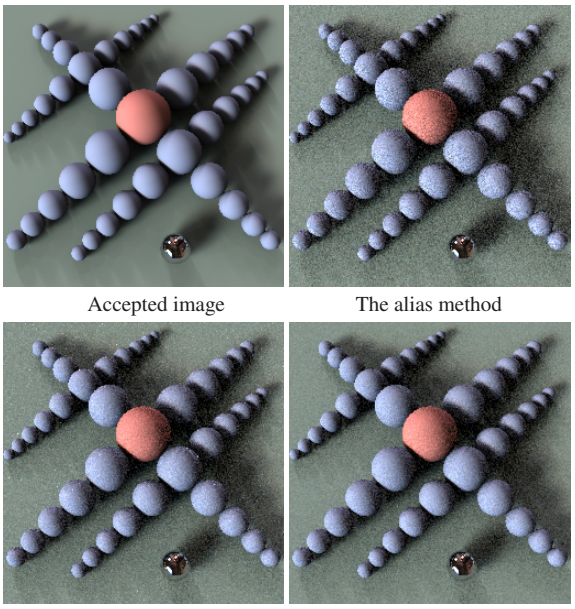
[KCODL06] KOPF J., COHEN-OR D., DEUSSEN O., LISCHINSKI D.: Recursive wang tiles for real-time blue noise. *ACM Transactions on Graphics (Proceedings of*

© The Eurographics Association and Blackwell Publishing 2008.



(a) (b) (c) (d)

Figure 9: *Transformed point sets and their Fourier transforms. (a) pdf and input point set, (b) the alias method, (c) approximate cdf inversion, (d) binary and linear search in the cdf.*



Accepted image | The alias method

Approximate cdf inversion | cdf search

Figure 10: *Environment map lighting with 16 samples per pixel using the different inversion techniques. All renderings transform a Hammersley point set to the environment map distribution.*

SIGGRAPH 2006) 25, 3 (2006), 509–518.

[LD06] LAGAE A., DUTRÉ P.: *A Comparison of Methods for Generating Poisson Disk Distributions*. Report CW 459, Department of Computer Science, K.U.Leuven, Leuven, Belgium, August 2006.

[LRR05] LAWRENCE J., RUSINKIEWICZ S., RAMAMOORTHI R.: Adaptive numerical cumulative distribution functions for efficient importance sampling. In *Eurographics Symposium on Rendering* (June 2005).

[Ost07] OSTROMOUKHOV V.: Sampling with polyominoes. *ACM Transactions on Graphics (Proceedings of SIGGRAPH 2007) 26*, 3 (2007), 78.

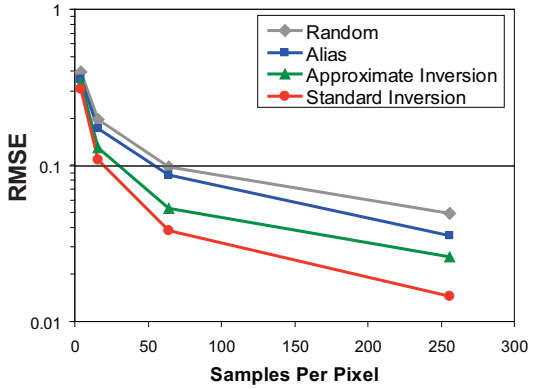


Figure 11: *Root mean squared error (RMSE) for the scene in figure 10, rendered with different inversion techniques and samples per pixel. "Random" shows the results of standard inversion using a random point set as input to the transformation routine. The other renderings use a Hammersley point set as input.*

[RSH05] RESHETOV A., SOUPIKOV A., HURLEY J.: Multi-level ray tracing algorithm. In *ACM Transactions on Graphics (Proceedings of SIGGRAPH 2005)* (2005), ACM Press, pp. 1176–1185.

[TCE05] TALBOT J. F., CLINE D., EGBERT P. K.: Importance resampling for global illumination. In *Eurographics Symposium on Rendering* (2005), Eurographics Association, pp. 139–146.

[Vos91] VOSE M.: A linear algorithm for generating random numbers with a given distribution. In *IEEE Transactions on Software Engineering* (September 1991), vol. 17(9), pp. 972–975.

[Wal77] WALKER A.: An efficient method for generating discrete random variables with general distributions. In *ACM Transactions on Mathematical Software* (September 1977), vol. 3(3), pp. 253–256.

[WSBW01] WALD I., SLUSALLEK P., BENTHIN C., WAGNER M.: Interactive rendering with coherent ray tracing. In *Proceedings of Eurographics 2001*, vol. 20(3). Blackwell Publishing, 2001, pp. 153–164.

Appendix A: Integer cdf inversion with linear search.

Here we sketch a proof that the linear search in section 3.1 runs in constant time on average:

Given that F and $\hat{F}^{-1}$ are the same size, the average difference between consecutive entries in $\hat{F}^{-1}$ must be no more than 1. Furthermore, a search within F must range between consecutive entries in $\hat{F}^{-1}$, inclusively. Therefore, assuming uniform input points, the average number of checks performed during a search must be no more than 2, and the search runs in constant time on average.

© The Eurographics Association and Blackwell Publishing 2008.