Implementing Bridson's Fast Poisson Disk Sampling in PBRT

Yuting Ye*

Georgia Institute of Technology

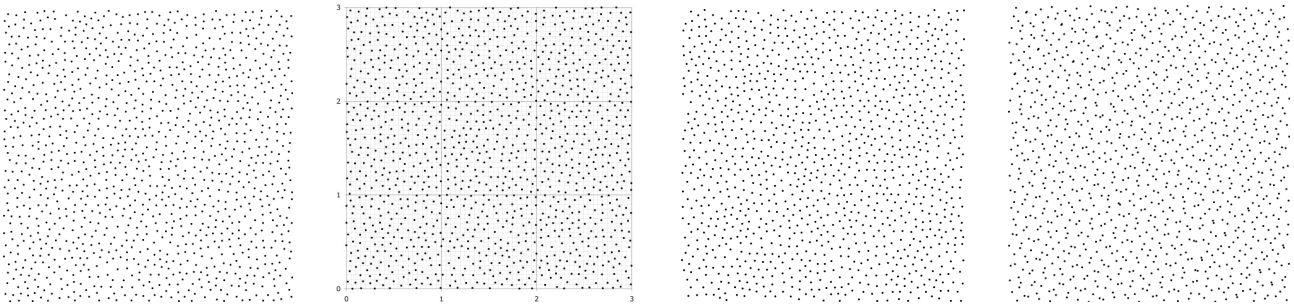

Figure 1: Sampling patterns. From left to right is Poisson disk pattern, same poisson disk pattern with pixel boundary and background grid embedded, best candidate pattern and low discrepancy pattern

1 Introduction

Sampling distributions with blue noise characteristic are useful in computer graphics. Rendering, especially distribution ray tracing, relies on the sampling patten to capture many high dimensional continuos functions. Poisson disk distribution, where all the samples are at least distance r apart, is considered ideal for rendering purpose [Cook 1986] also because it mimics the distribution of photoreceptors in a primate eye [Yellott 1983]. However, it was too expensive to generate Poisson disk pattern in runtime by naïve dart throwing. Best candidate algorithm [Mitchell 1991] provides a termination guarantee by choosing the best one among candidate samples but does not guarantee true Poisson disk distribution. Dunbar and Humphreys overcomes this problem in $2D$ by carefully maintaining a "scalloped sector" data structure that encodes the valid sampling region [Dunbar and Humphreys 2006]. Their algorithm can run in linear time and guarantees maximal Poisson disk distribution. Bridson further overcomes this problem in arbitrary dimensions by dart throwing in possible local regions [Bridson 2007]. His approach also runs in linear time and is very simple to implement.

This work implements Bridson's fast Poisson disk sampling algorithm in PBRT (Physically Based Ray Tracer [Pharr and Humphreys 2004]) and compares the result with best candidate algorithm and low discrepancy sampling. Although Bridson already distributed the example code for general purpose, there are specific rendering related issues that worth discussing.

2 Algorithm

Bridson's algorithm samples only inside the annulus between r and $2r$ of a given sample. Dart throwing is used to discover possible new samples, which will be compared with all neighbor samples in the background grid and then kept in an active list if valid. When no sample point can be added around the given sample after maximum attempts, this sample is removed from the active list. Background grid is implemented as an n dimensional array with cells that can contain at most one sample. Given desired distance r in n dimension space, the diagonal $L2$ distance of a grid cell with length l is nl^2, thus we choose $l = r/\sqrt{n}$. Since each sample is generated in constant time, this algorithm runs in linear time. Although maximal coverage depends on the number of attempts in dart throwing, this factor is only affected by dimension n and is constant in the algorithm. In practice this number is usually small (Bridson proposed 30 in $3D$ case). The following procedure describes the algorithm.

proc SAMPLE
    generate the first random point p_0
    insert p_0 into active list
    place the index of p_0 (zero) to the background grid cell
    while active list is not empty **do**
        choose a random point p from active list
        for attemp = 1:maxAttemp
            get new sample p' around p between r and $2r$
            for each non-empty neighbor cell around p'
                if p' is closer than r
                    break
                end if
            end for
            if p' is far from all neighbors
                insert p' to active list
                place the index of p' to background grid cell
                break
            end if
        end for
        if maxAttemp exceed
            remove p from active list
        end if
    end while
    return

3 Implementation

This algorithm is implemented as a sampler plugin in PBRT. Sampler is used to generate camera rays and integrator samples. The camera ray is a $5D$ vector that samples the image plane, camera lens and time respectively. Although this algorithm is capable of sampling this $5D$ space without extra cost, full coverage of each of these three domains is more desirable in rendering. Therefore, image plane, lens and time are sampled separately and then randomly associated to form the camera ray.

*email: yuting@cc.gatech.edu

Lens and time samples are in the range of $[0,1]$ for each dimension when the image plane samples range in the image size. For antialiasing purpose, we want each pixel to cover the whole lens and time domain. It would be better to sample each pixel separately rather than the image plane as a whole. This is also potentially more efficient in memory usage. Moreover, we only need a background grid that covers one single pixel and discard the samples when done with tracing this pixel. To make sure samples at the pixel boundary do not cluster together, we need to keep previous samples. With sampling order from left to right and up to down, rightmost samples from the previous pixel and the bottom samples from pixels above need to be kept and compared to leftmost and topmost samples of the current pixel. Up to 2 rows and columns at the boundary of background grid need to keep in $2D$ case because $r = \sqrt{2}l$. For n dimension, this number will be the ceiling of $\sqrt{n}$ in each dimension. Note that now the first sample in each pixel, except for the first one, should also compare to adjacent samples in case it is at the boundary.

One major problem we encounter is that the number of samples generated for each pixel is hard to control precisely. In $2D$ case, expected distance between two samples is $\sqrt{2.5}r$, which means each sample covers an area of $5\pi r^2/8$. Given desired samples per pixel N, r is calculated as

$$r = \sqrt{8/(5\pi N)}$$

However the actual number of samples differs every time for every pixel. Even the total number of samples for the whole image diverge from the expected value because of rounding error. Since pixel sample, lens sample and time sample are generated separately, their numbers do not always match. As a result, we have to take the smallest number to associate the camera ray and discard some samples, which in turn creates holes in the sampling domain. This effect is apparent in the final image. For the same reason, it is not practical to generate samples for the integrator given desired number of samples, even though this algorithm is good at sampling high dimensional functions.

4 Results

From the sample pattern generated (Figure 1) we can see that the per pixel sampling method respects the boundary of adjacent pixels and maintains the overall Poisson disk pattern. This implementation is compared with the built in best candidate sampler and low discrepancy sampler in PBRT on an area light scene (Figure 2), a simple depth of field scene (Figure 3, Figure 4) and a classic checkerboard texture (Figure 5, Figure 6). Figure 1 shows that best candidate algorithm has similar pattern to Poisson disk when low discrepancy pattern cannot avoid clustering. However rendering results indicate that low discrepancy works better than Poisson disk in some cases. In contrast, although best candidate has similar pattern to Poisson disk, its results are generally noisier. One reason could be because PBRT always reuses a pre-generated pattern to tile over the whole image plane for best candidate algorithm is too slow in runtime. Another reason is this Poisson disk implementation usually generates more samples than the expected value.

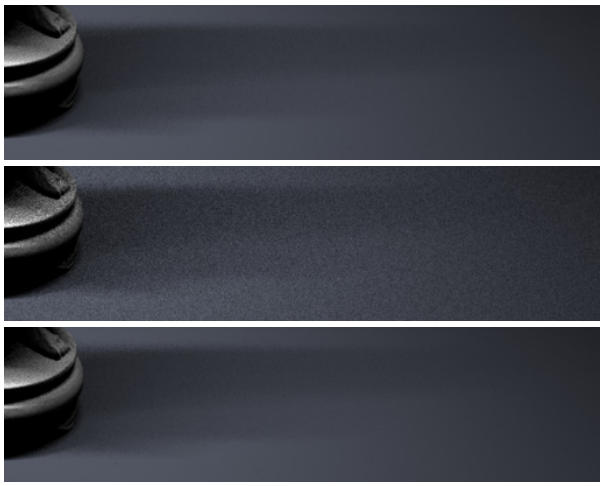

Figure 2: Budda figure lit by an area light on top with 4 samples per pixel. From top to buttom is Poisson disk, best candidate and low discrepancy sampling pattern

Figure 3: Depth of field with 16 samples per pixel. From left to right is Poisson disk, best candidate and low discrepancy sampling pattern.

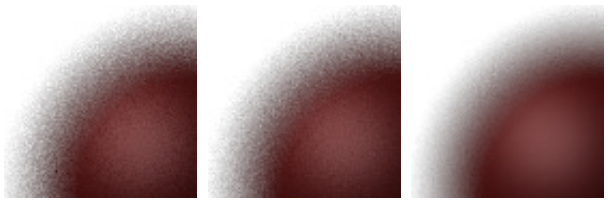

Figure 4: Depth of field with 64 samples per pixel. From left to right is Poisson disk, best candidate and low discrepancy sampling pattern.

References

BRIDSON, R. 2007. Fast poisson disk sampling in arbitary dimensions. *SIGGRAPH Sketches Program.*

COOK, R. L. 1986. Stochastic sampling in computer graphics. *ACM Trans. on Graphics (SIGGRAPH).*

DUNBAR, D., AND HUMPHREYS, G. 2006. A spatial data structure for fast poisson-disk sample generation. *ACM Trans. on Graphics (SIGGRAPH).*

MITCHELL, D. P. 1991. Spectrally optimal sampling for distribution ray tracing. *SIGGRAPH*.

PHARR, M., AND HUMPHREYS, G. 2004. *Physcially Based Rendering*: *From Theory to Implementation.* Morgan Kaufmann, August.

YELLOTT, J. I. 1983. Spectral consequences of photoreceptor sampling in the rhesus retina. *Science 221*, 382–385.

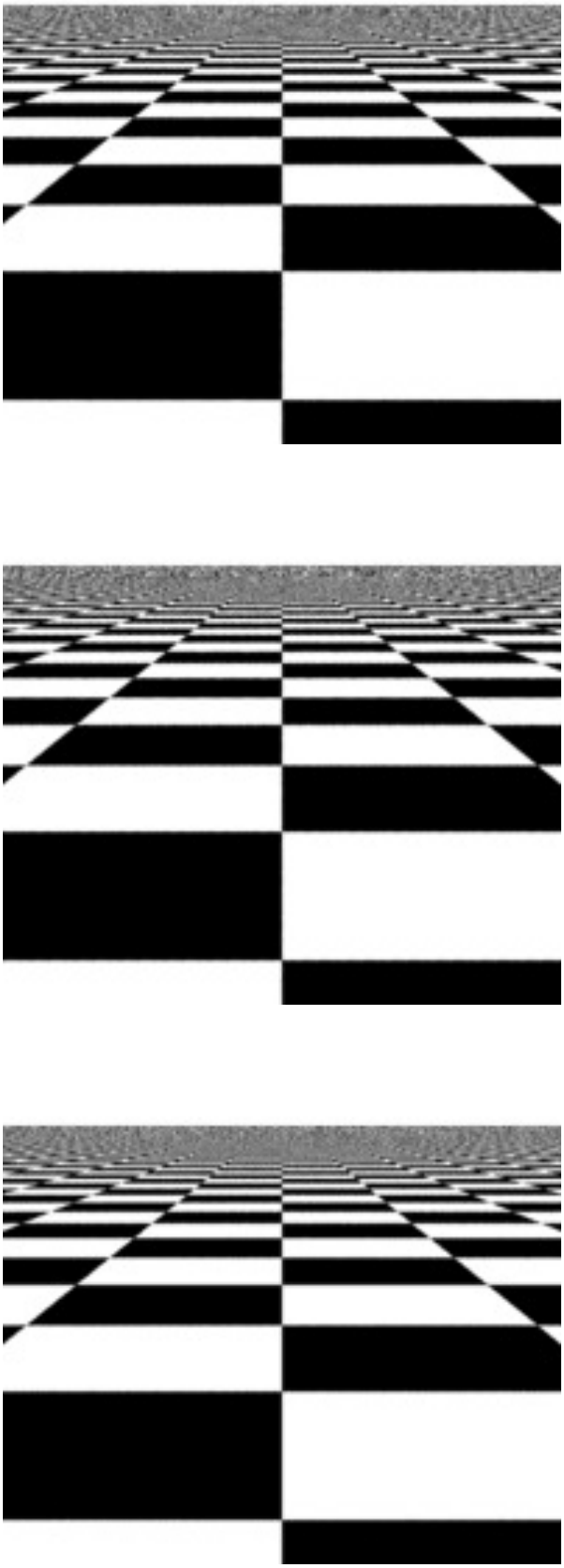

Figure 5: Checkerboard texture with 16 samples per pixel. From top to buttom is Poisson disk, best candidate and low discrepancy sampling pattern

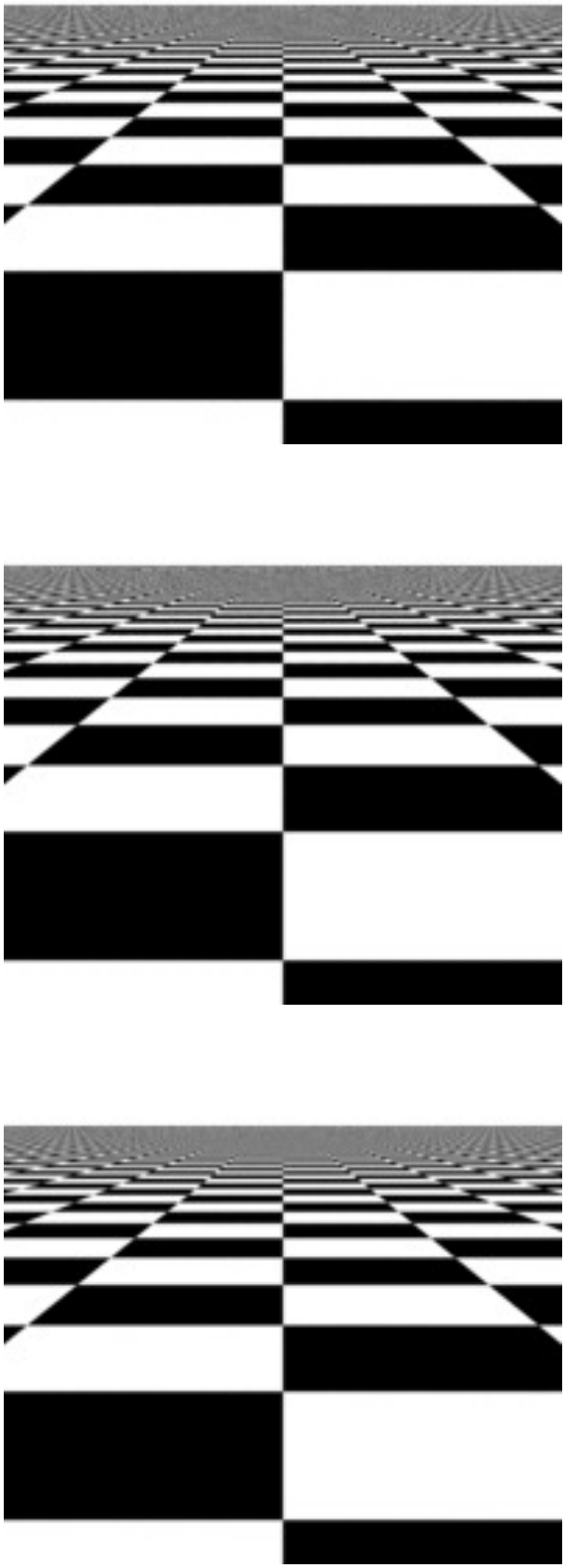

Figure 6: Checkerboard texture with 64 samples per pixel. From top to buttom is Poisson disk, best candidate and low discrepancy sampling pattern