


Stochastic Lightcuts – Supplemental Document

Cem Yuksel 

University of Utah, UT, USA

This supplemental document provides additional implementation details for the stochastic lightcuts method and includes full-resolution versions of the images in the paper.

Implementation Details

The stochastic lightcuts method can be easily implemented on an existing implementation of lightcuts.

The light tree construction algorithm requires no change. Stochastic lightcuts can use the same light tree as lightcuts. Alternatively, a deterministic light tree construction algorithm can be used. Since we use stochastic sampling during lighting estimation, there is no need to introduce a stochastic process in light tree construction to avoid bias. The quality of the light tree merely determines the amount of noise in the final outcome and, thereby, the convergence rate.

The tree storage, however, requires some minor modifications. First of all, there is no need to store a representative light per internal node. While a simplistic implementation of dead branches can terminate the light tree traversal by returning the representative light, it would be more efficient to not return a light at all. This is because no light within a dead branch can have a non-zero illumination contribution.

Each internal node must store the total intensity of lights within its subtree. The difference from the original lightcuts method, which must also store the total intensity under an internal node, is that the total intensity of lights within a subtree is only used for computing the probability of selecting the node or its sibling (i.e. w_1 and w_2). Therefore, it is sufficient to simply store a scalar value $\|\mathbf{I}_j\|$ and another scalar value $\|\mathbf{I}_j^D\|$ for supporting directional light sources or other light types without inverse-square attenuation.

Algorithm 1 shows the pseudocode of the light selection algorithm using the hierarchical importance sampling approach of stochastic lightcuts. Note that this algorithm either returns *false*, when the given node j is a dead node, or returns the light sample and the probability of selecting it within the given subtree. Notice that Algorithm 1 can traverse the entire light tree when required. Therefore, it can introduce additional computation cost, as compared to the original lightcuts method, which simply returns a pre-selected representative light.

Note that dead branch avoidance in Algorithm 1 leads to biased light sampling.

Algorithm 1: Pseudocode of selecting a light within a light subtree using hierarchical importance sampling.

```

1 function SelectLight ( $j$ )
   //  $j$  is the root node of a light subtree
2    $p \leftarrow 1$  // initialize the probability of picking the light
3    $r \leftarrow$  a random value between 0 and 1
4   while node  $j$  is not leaf do
5      $w_1 \leftarrow$  the weight of the first child node
6      $w_2 \leftarrow$  the weight of the second child node
7     if  $w_1 + w_2 > 0$  then
8        $p_1 \leftarrow w_1 / (w_1 + w_2)$ 
9       if  $r < p_1$  then
10        if  $p_1 < 1$  then
11          Push the second child node of  $j$  to the
            stack with  $p$  and  $r$ .
12        end
13         $p \leftarrow p \cdot p_1$  // update the probability
14         $r \leftarrow r / p_1$  // rescale the random value
15         $j \leftarrow$  the first child node of  $j$ 
16      else
17        if  $p_1 > 0$  then
18          Push the first child node of  $j$  to the stack
            with  $p$  and  $r$ .
19        end
20         $p \leftarrow p \cdot (1 - p_1)$  // update the probability
21         $r \leftarrow (r - p_1) / (1 - p_1)$  // rescale  $r$ 
22         $j \leftarrow$  the second child node of  $j$ 
23      end
24    else
25      // Dead branch detected.
26      if the stack is not empty then
27        Pop  $j$ ,  $p$ , and  $r$  from the stack
28      else
29        return false // no light sample found
30      end
31    end
32   $i \leftarrow$  the light sample of the leaf node  $j$ 
33  return ( $i, p$ )

```

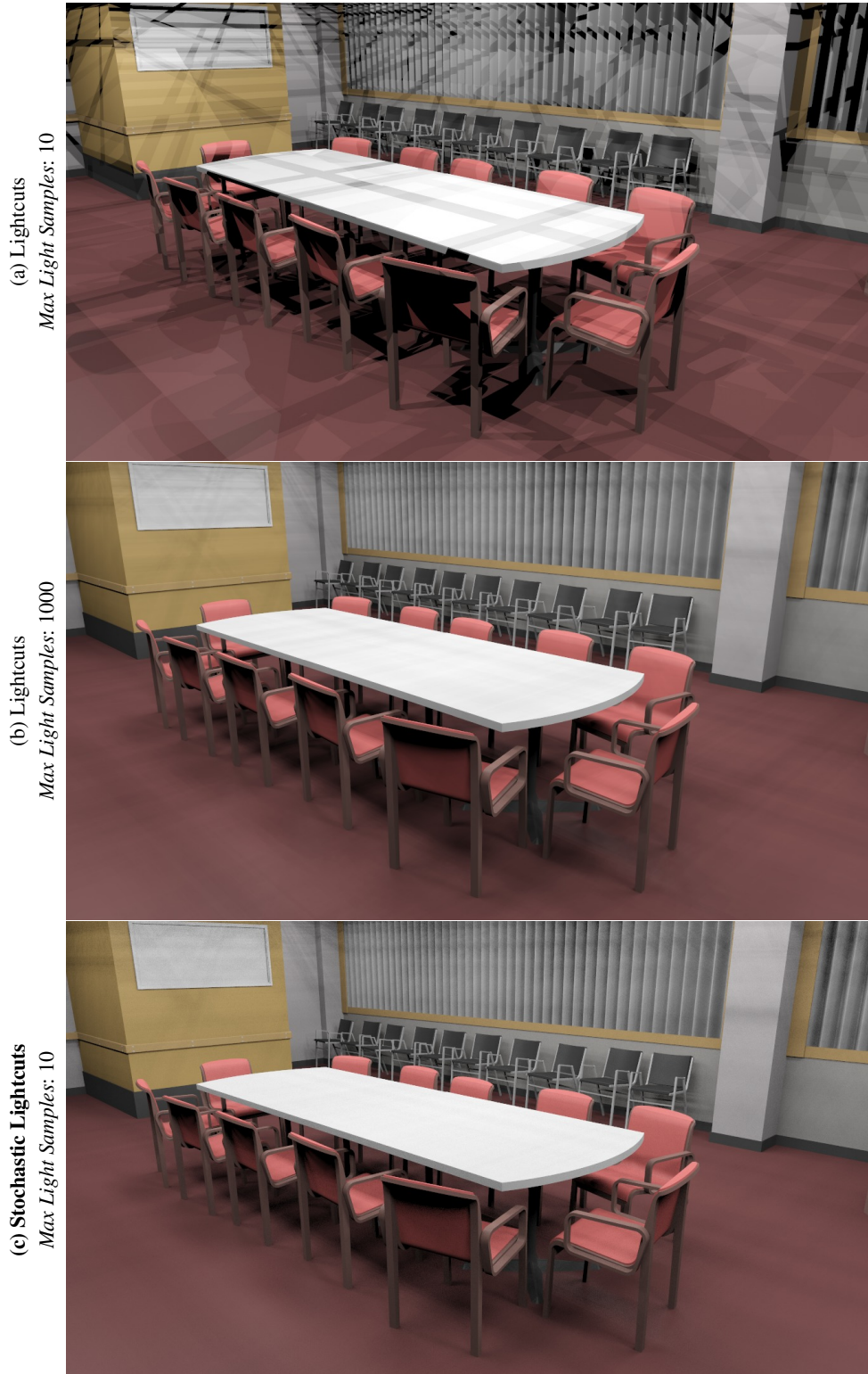


Figure 1: Conference scene with direct illumination from 14 light fixtures (on the ceiling), each containing 100 light sources, rendered using 64 samples per pixel. (a) Lightcuts with up to 10 light samples produces prominent stripes on the table, on the floor, and in the background, due to the shadows of the light fixture details. (b) When up to 1000 light samples are permitted with lightcuts, the stripes become less noticeable in a still frame, but such artifacts still persist and lead to substantial flickering in animations, and it also takes almost 30× longer time to render. (c) Our stochastic lightcuts method achieves a low-noise solution with no visible artifacts using only 10 light samples per lighting estimation.

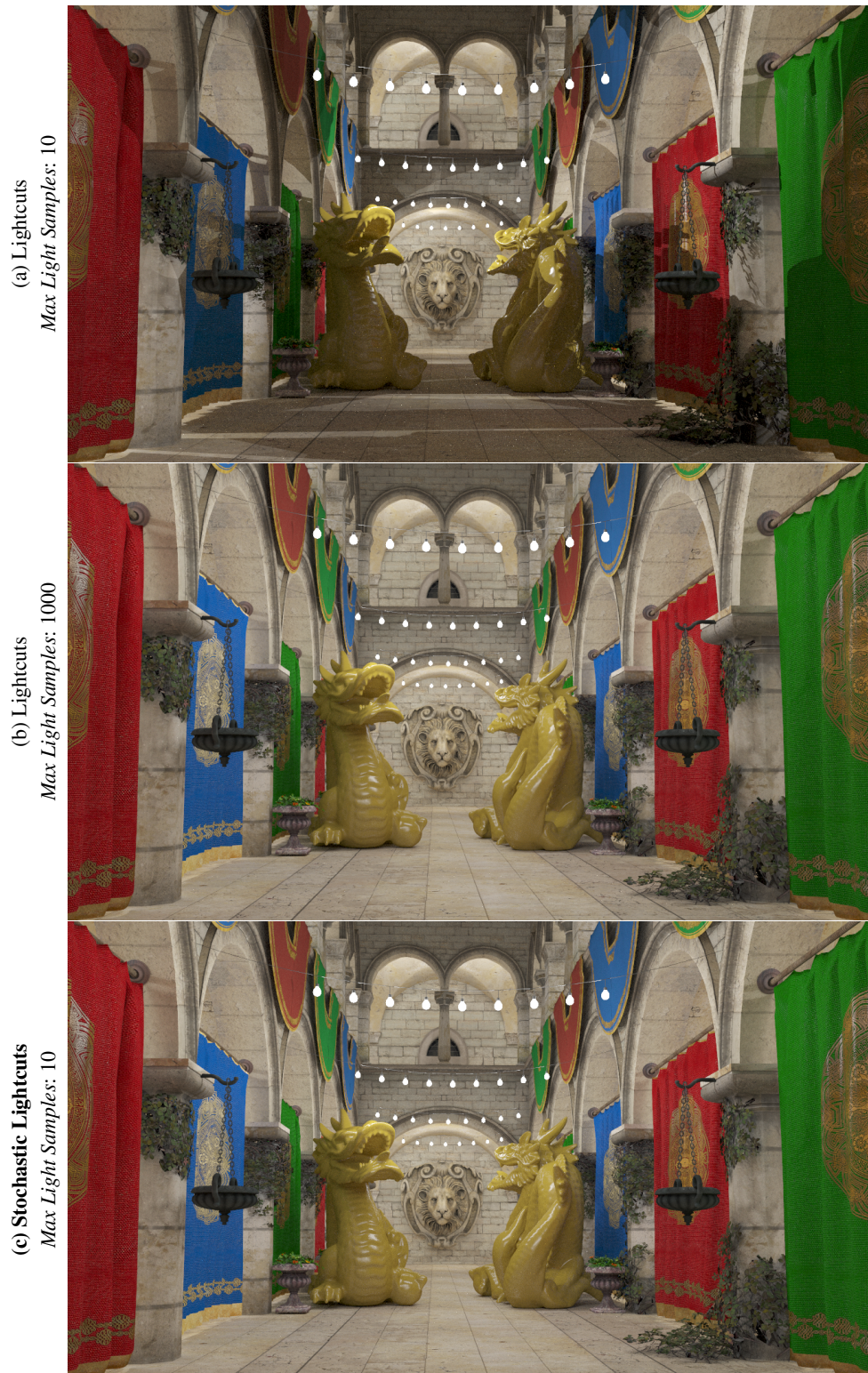


Figure 2: Crytek Sponza scene with direct illumination from 1644 light sources, rendered using path tracing with 5 bounces and 64 samples per pixel. (a) Lightcuts with up to 10 light samples produces a substantial amount of error and correlation artifacts. (b) Using up to 1000 light samples reduces the error, but still leads to visible flickering and takes more than 20× render time. (c) Our stochastic lightcuts method can produce a fast, temporally-stable, and low-noise lighting estimation with up to 10 samples.

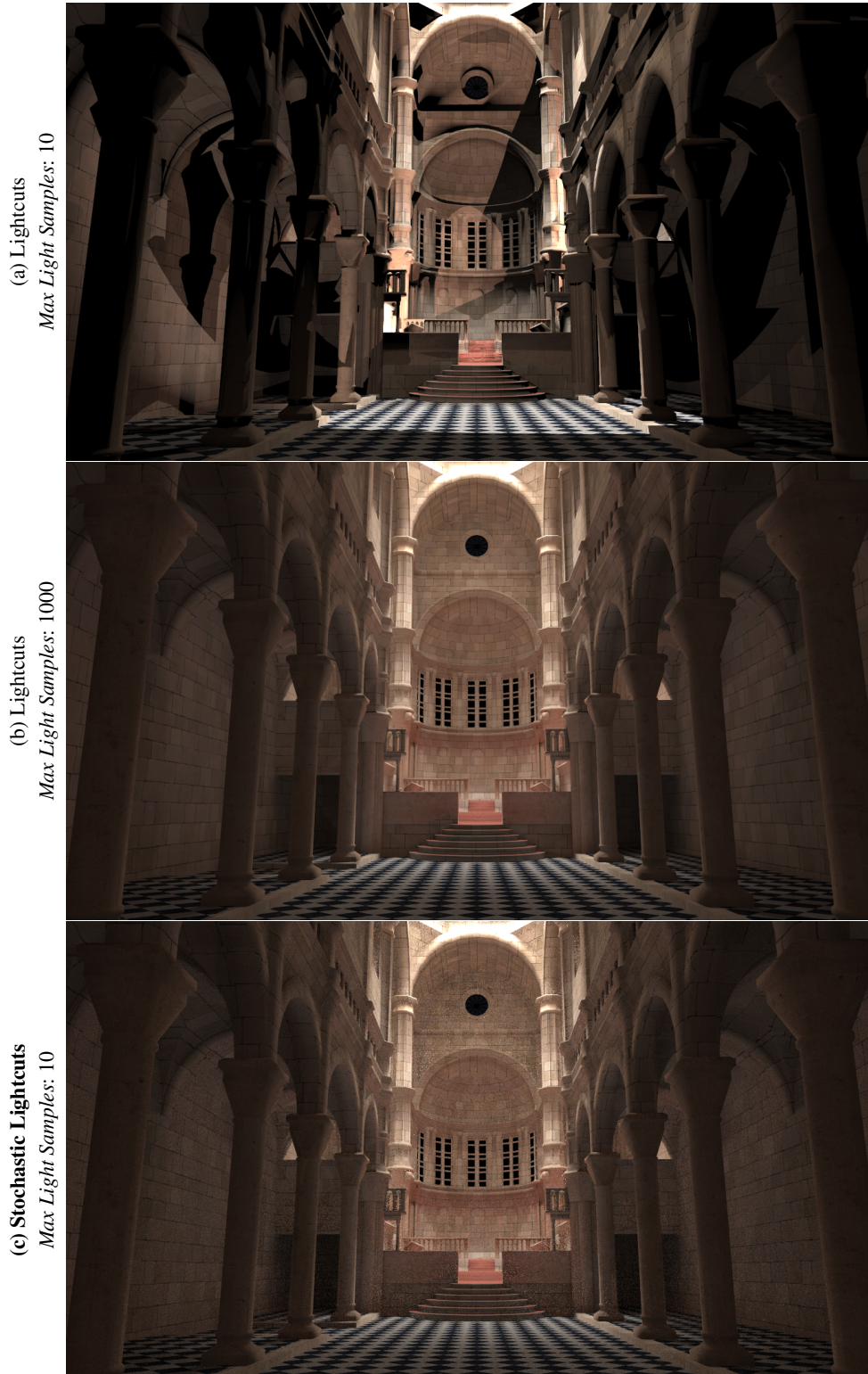


Figure 3: Sibenik scene illuminated by one million virtual spherical lights, generated with up to 8 bounces from 120 light sources near the top of the dome, rendered using 64 samples per pixel. (a) Lightcuts with up to 10 light samples produces severe visual artifact in the form of sharp shadow lines. (b) Using up to 1000 light samples with lightcuts makes these artifacts less prominent, but the result still flickers in animation and the render time becomes about 80× longer. (c) Our stochastic lightcuts method produces a temporally-stable solution with only 10 light samples.

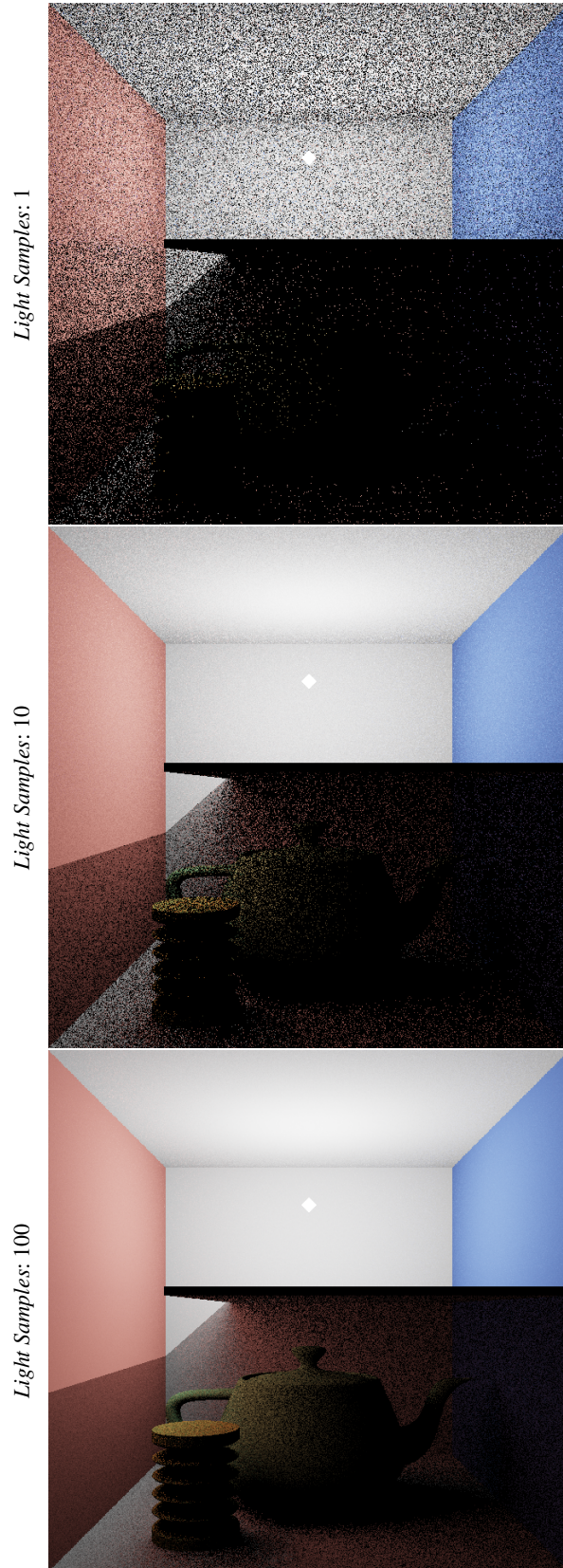


Figure 4: Traditional importance sampling [SWZ96].

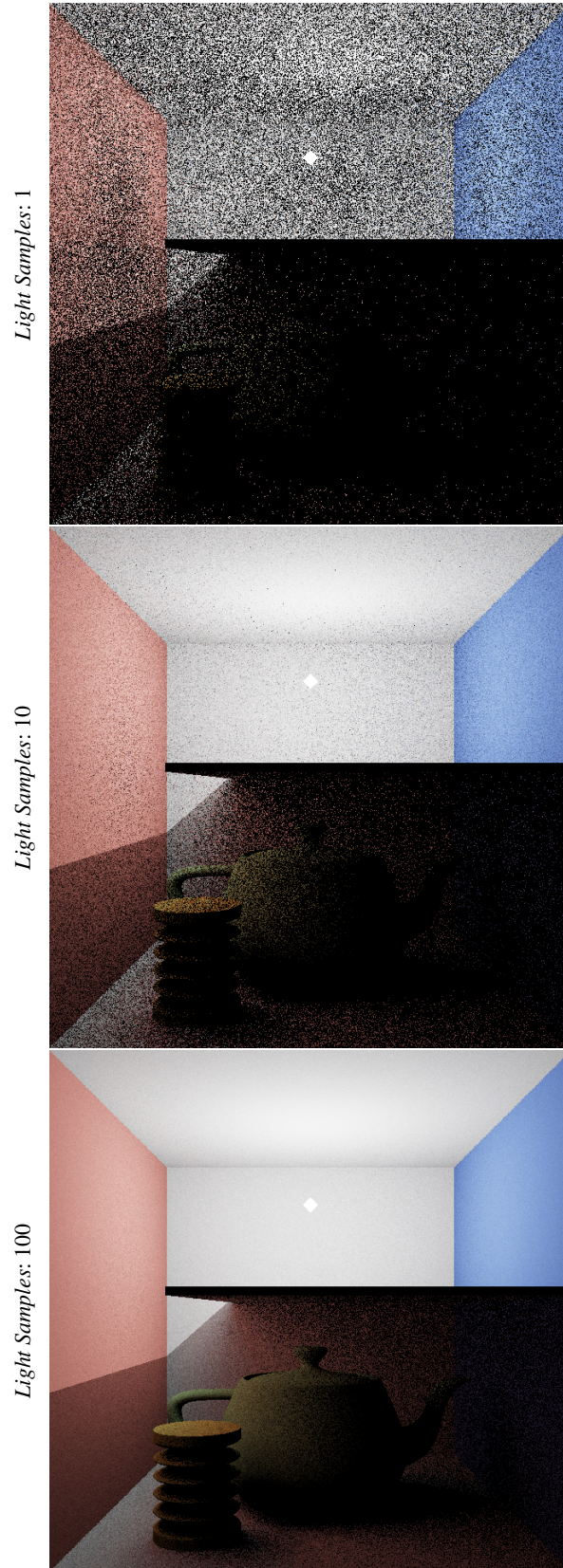


Figure 5: Adaptive tree splitting [EK18].

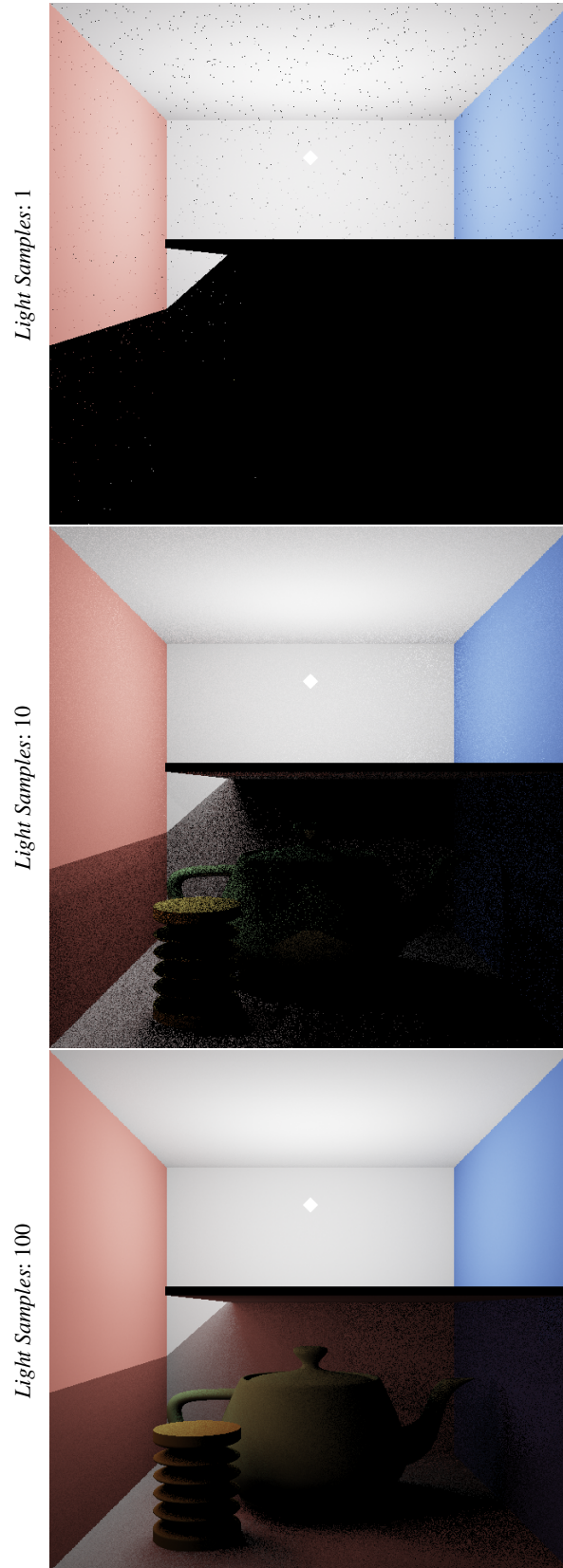


Figure 6: Multidimensional lightcuts [WABG06].

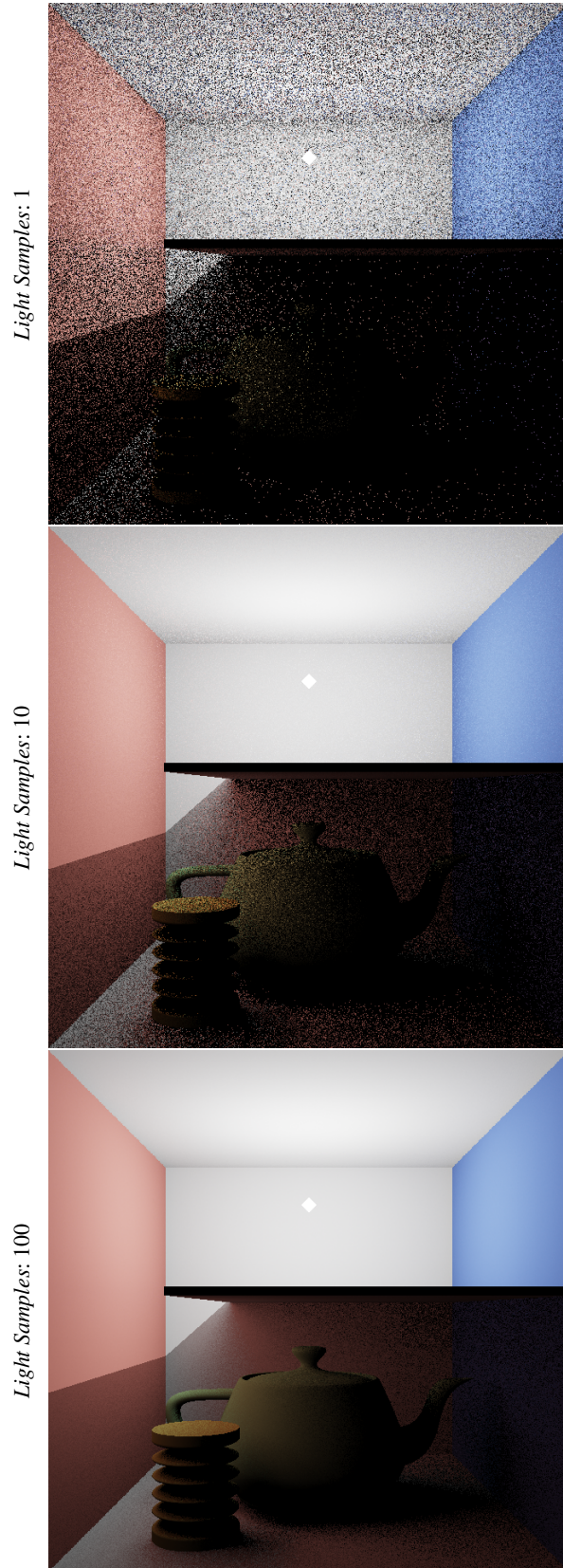


Figure 7: Stochastic lightcuts.

References

- [EK18] ESTEVEZ A. C., KULLA C.: Importance sampling of many lights with adaptive tree splitting. *Proc. ACM Comput. Graph. Interact. Tech. (Proceedings of HPG 2018)* 1, 2 (2018), 25:1–25:17. [6](#)
- [SWZ96] SHIRLEY P., WANG C., ZIMMERMAN K.: Monte carlo techniques for direct lighting calculations. *ACM Transactions on Graphics* 15, 1 (1996), 1–36. [5](#)
- [WABG06] WALTER B., ARBREE A., BALA K., GREENBERG D. P.: Multidimensional lightcuts. *ACM Transactions on Graphics (Proceedings of SIGGRAPH 2006)* 25, 3 (2006), 1081–1088. [7](#)