Contents lists available at ScienceDirect

Computers & Graphics

journal homepage: www.elsevier.com/locate/cag

Interactive Rendering of Translucent Materials under Area Lights using Voxels and Poisson Disk Samples

Ming Di Koa[a,*], Henry Johan[b], Alexei Sourin[a,b]

[a]*School of Computer Science and Engineering Nanyang Technological University, Singapore*
[b]*Fraunhofer Singapore, Singapore*

ARTICLE INFO

Article history:
Received December 27, 2017

Keywords: Translucent Materials, Area Lights, Direct Illumination, Indirect Illumination, Interactive Rendering, Virtual Worlds

ABSTRACT

Interactive rendering of translucent materials in virtual worlds has always proved to be challenging. Rendering their indirect illumination produces further challenges. In our work, we develop a voxel illumination framework for translucent materials illuminated by area lights. Our voxel illumination uses two existing voxel structures, the Enhanced Subsurface Light Propagation Volumes (ESLPV), which handles the local translucent material appearance and the Light Propagation Volumes (LPV), which handles indirect illumination for the surrounding diffuse surfaces. By using a set of sparse translucent Poisson disk samples (TPDS) and diffuse Poisson disk samples (DPDS) for the ESLPV and LPV, illumination can be gathered from area lights effectively. This allows the direct illumination of the translucent material to be rendered in the ESLPV, and the diffuse indirect illumination of the surrounding scene can be rendered in the LPV. Based on experiments, a small number of Poisson disk samples in each voxel are sufficient to produce good results. A uniform set of Poisson disk samples on the translucent objects is resampled and chosen as Translucent Planar Lights (TPLs) and is used to distribute lighting from translucent objects into the LPV by an additional gathering process. Our technique allows for direct and indirect illuminations from highly scattering translucent materials to be rendered interactively under area lighting at good quality. We can achieve similar effects, such as low-frequency scattered light illumination from translucent materials, when compared to offline renderers without precomputations.

1. Introduction

Interactive rendering of scenes with highly scattering translucent objects is difficult to compute efficiently. Although approximation techniques can model the local illumination of translucent materials, techniques that model their indirect illumination interactively on surrounding surfaces are rarely researched. Translucent objects can act as area lights when they release scattered light to their surroundings to produce visual effects in the appearance as soft diffuse color bleeding. For example in Figs. 1a and 1b, a translucent glass cup can produce low-frequency scattered illumination and illuminate its surrounding plane. This effect is not present on diffuse surfaces as they are usually optically thick. For example, the diffuse glass cup in Fig. 1c) has no indirect illumination reaching the external plane as there are no possible light paths. Although it is easier for rendering algorithms to work with translucent materials using point light sources, area lights often produce soft shadows which exhibit coloration by scattered lighting within the translucent materials. This effect contributes greatly to the vi-

⋆

*Corresponding author: Ming Di Koa
e-mail: mdkoa1@e.ntu.edu.sg (Ming Di Koa)

sual realism in interactive rendering.

Monte Carlo techniques [1, 2] can render inter reflections from translucent materials but they are too computationally expensive. Improvements in other methods, such as photon mapping, allow translucent materials to produce indirect illumination effects [3, 4], but they are impractical for generating images at interactive rates. Current radiosity [5] approaches allow for real-time computations, but at the expense of long precomputation time and large storage data for precomputing form-factors. In interactive rendering, it is often important to reduce sampling costs while maintaining realism.

In this paper, we present a voxel based illumination approach that renders translucent objects under direct area light illumination and renders indirect illumination from translucent and diffuse surfaces interactively without precomputations. We present three main contributions in this work.

- A Poisson disk sampling solution to allow lighting information from area lights to be injected into a voxel structure for rendering translucent objects
- A Poisson disk sampling solution to allow lighting information from area lights to be injected into a voxel structure for rendering indirect illumination for diffuse surfaces.
- An interreflection framework for distributing indirect illumination from translucent objects to their nearby diffuse surfaces. This allows translucent objects to be treated as area lights.

Our proposed illumination pipeline allows for interactive rendering without any precomputations and achieves the soft diffuse color bleeding effect of scattered light from translucent materials similar to offline renderers.

2. Related Work

2.1. Interactive subsurface scattering

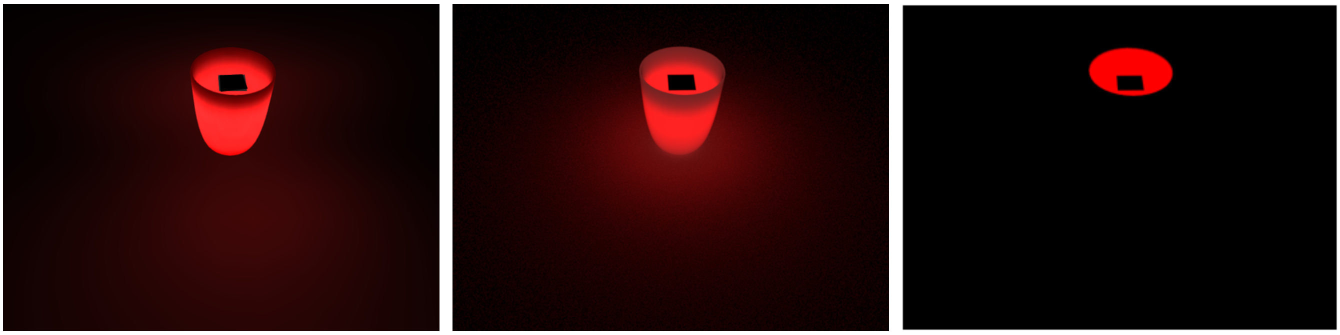
In recent years, several real-time techniques have been developed for translucent materials and its scattering phenomenon known as subsurface scattering. Lensch et al. [7] formulated the subsurface scattering effect as calculation of throughput between vertices using radiosity techniques. Mertens et al.[8] and Carr et al. [9] suggested using a hierarchy of clustered triangles and multi-resolution mesh respectively similar to hierarchical radiosity. Yu et al. [5] extended the radiosity method for subsurface scattering to support multiple bounce global illumination. However, radiosity approaches require a large number of form factors which may in turn require a large amount of computation as well as storage. Wang et al. [10] and Adam et al. [11] devised tetrahedralization techniques for discretizing a mesh into a 4-connected mesh structure. Their techniques are able to handle much more complex geometries, especially those with thin features. However, the time needed to process the mesh to a suitable structure, though automated, may still take minutes.

Texture space diffusion was introduced by Borshukov and Lewis [12] particularly for skin rendering. They suggested unwrapping the 3D mesh of the object into a 2D texture while storing the irradiance information onto this texture. The 2D texture, known as irradiance map, then undergoes a convolution process with a diffusion profile as the filtering kernel. The resulting texture after being blurred, produces a simulated subsurface scattering effect. Jimenez et al. [13] and d'Eon et al. [14] proposed simulating the filtering kernel with a set of weighted Gaussians. This allowed convolutions to be done quickly as Gaussians are symmetrically separable functions. However, texture space methods require mesh parameterization which might require some preprocessing time. Jimenez et al. [15], [16] overcame the limitations of texture space rendering by using a screenspace approach. However, screenspace approaches lack irradiance information on the back of the object. Echevarria et al. [17] and Munoz et al. [18] proposed hybrid methods involving screenspace and texture space approaches. They enclose the object with a set of equally spaced planes perpendicular to the viewing direction. They compute the sum of inter-plane radiance contributions of each plane by performing convolutions with a distance based varying size diffusion kernel. Their results produce convincing translucent effects but suffer from temporal artifacts from changing viewpoints due to inconsistent irradiance information from the changing geometry of the irradiance planes.

Geist et al. [19] proposed using a voxel-based approach to simulate a light propagation process. Their process was much more storage efficient but computationally expensive. Bernabei et al. [20] proposed reducing the principal directions of propagation to six directions which enables rendering to be done at interactive rates. Borlum et al. [21] reformulated the LPV technique, and changed the LPV into a Subsurface Light Propagation Volumes (SSLPV). Unlike the LPV, the SSLPV is a voxelized structure of an object and not a scene. The flux is injected into the material and distributed by a propagation process. An extension using a hierarchical propagation approach was proposed by Koa et al.[22]. Their approach, the ESLPV, allows light to scatter across further distances simulating highly scattering materials. However, these works deal with local illumination models and do not extend to indirect illumination.

2.2. Interactive indirect illumination

Rendering dynamic indirect lighting requires many ray-triangle intersection tests. Many algorithms work on determining whether a pixel should be illuminated based on adaptive algorithms that simplify either the scene or the ray tracing process. In recent years, emphasis has been given to algorithms that work on voxel-based illumination to compute dynamic illumination on the fly. Crassin et al. [23] proposed an indirect illumination algorithm that uses a sparse voxel octree (SVO) to store illumination and geometric information in its voxels. The SVO structure allowed an efficient cone tracing solution to reduce the costs of ray tracing, as well as to generate filtered mipmaps which allows for blending of illumination information across different resolutions. However, it requires a large memory space for pre-allocating voxel information, which is usually larger than the acceleration structure used for ray tracing. Sugihara et al. [24] proposed using a layered shadow map to store illumination in the Reflective Shadow Maps (RSM) [25] while



(a) Our result (b) Mitsuba Path Tracer Global Illumination (c) Diffuse material

Fig. 1. Rendering of a water glass with a red scattering material illuminated with a single area light source inside the cup at 1024x768. We show our rendering in (a) with both the direct and indirect illumination components, and a reference image generated by Mitsuba Path Tracer (64 samples per pixel) in (b). The translucent material in the reference image is rendered with Photon Beam Diffusion method [6]. Image (a) is rendered in 146 ms. Image (b) is rendered in 54.96 seconds. Image (c) shows the same experiment setup but with a diffuse water glass. No light is scattered to the exterior of the water glass.

voxel structures continue to hold geometric information. This approach significantly reduces memory consumption. Crassin et al. [26] devised a combination of both voxels and precomputed radiance transfer (PRT) for compressing irradiance information into basis functions. Similarly, Iwanicki and Sloan [27] suggested generating multiple sampling points on the object and precomputing the incoming lighting at those points. PRT methods allow handling of occlusion and also illumination changes in run-time. The PRT based methods can be incorporated into the light baking features of existing game engines at the cost of some precomputation time. Kaplanyan et al. [28] proposed the Light Propagation Volumes (LPV) as an extremely memory efficient solution for single and multi-bounce indirect diffuse illumination. The LPV uses 4-Spherical Harmonics coefficients for each color channel to represent the flux distribution in each voxel. The initial flux distribution is defined using a RSM. The position, normal and flux density of each texel obtained in the RSM are injected into the corresponding voxel in the LPV. The LPV then undergoes a propagation process, where each step of the propagation process allows the flux distribution in each voxel to be distributed to its neighbouring voxel. After the flux distribution in the LPV has reached equilibrium, the final voxel results can be used as a representation of indirect diffuse illumination. Unfortunately, using the RSM restricts the LPV to only point and spot lights. Hedman et al. [29] proposed using a dynamic distribution of point lights that contributes to the visible pixels for each frame. In each frame, visible point lights can be re-used and new ones can be created. A heuristic sample distribution is defined to obtain temporal stability. In our work, we use the LPV approach with a fixed sparse set of Poisson-disk samples which achieves temporal stability and speed at the expense of accuracy.

3. Our pipeline

We combined the ESLPV, in Koa et al.[22] and LPV [28] for our illumination pipeline. The ESLPV renders local subsurface scattering effects while the LPV distributes the indirect illumination from diffuse surfaces as well as translucent objects. The 3D objects in our scene and the translucent objects are voxelized for the LPV. The translucent objects can be voxelized according to the solid voxelization algorithm described in Schwarz et al. [30]. Voxelization allows us to store geometric information and material properties into these voxels, which are used for propagation. In our work, we use the LPV structure from Doghramachi's work [31] because of the way geometric blockers are represented in the voxels. Their method of storing normals of blockers into a tetrahedron face allows us to extract the closest normals relevant to the surface we are rendering. This allows more accurate illumination computations.

As previous work for the ESLPV [22] and LPV [28] did not deal with area light illumination, we design a complete framework for rendering translucent materials under area lights as shown in Fig. 2. Our system starts off with area light and scene information (e.g., mesh information). We generate Poisson disk samples [32] for both our translucent and diffuse objects. The Poisson disk sampling algorithm produces an ideal distribution with a minimum Euclidean distance between samples. Subr et al. [33] work on Fourier analysis indicates that Poisson disk samples have lower variance than jittered and stratified sampling when sample quantity are sparse. We provide further comparisons in Section 5. We use the notation diffuse Poisson disk samples (DPDS) and translucent Poisson disk samples (TPDS) to describe the two sets of Poisson disk samples generated on diffuse and translucent surfaces respectively. Every TPDS will have its light intensity computed and transferred into the ESLPV for injection. The ESLPV renders the translucent object with the given set of TPDS. This set of TPDS is downsampled to a reduced set of translucent planar lights (TPLs). This set of TPLs is used to represent the translucent object as an area light source. The intensities of the TPLs can be obtained by sampling the ESLPV voxels with a rendering equation after the propagation stage has been completed.

Next, we compute the reflected flux at the location of each DPDS. The reflected flux is computed with both the area light information and the TPLs, indicating that each DPDS now contains the reflected flux of direct illumination from the area light and the TPLs. The DPDS are injected by accumulating their intensities with previous TPLs into the LPV and rendered. By propagating the light intensity in the LPV designed

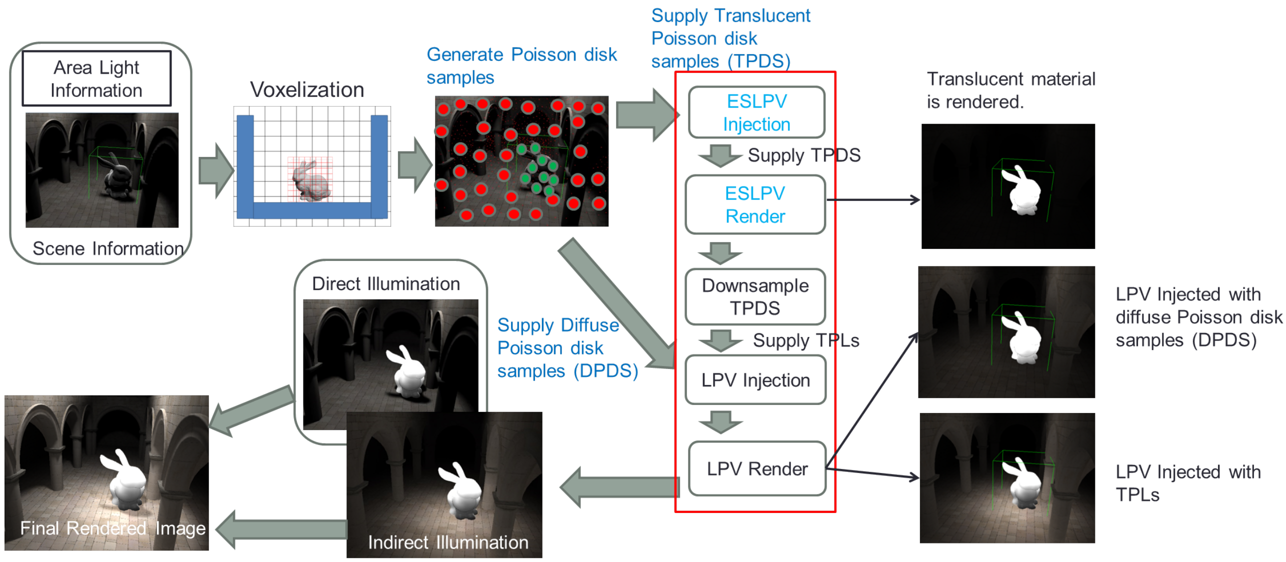


Fig. 2. The pipeline of our work with the indirect illumination combined with the direct illumination output.

by Doghramachi[31], we are able to simulate the indirect illumination of translucent materials as well as diffuse materials in the scene. The final rendered image is combined with our direct illumination (for diffuse surfaces), which handles area light direct illumination. We implemented the previously proposed method of direct illumination (Koa et al. [34]) using multi-resolution rendering under area lighting. This method is able to produce high quality soft shadows and is suitable for overlaying our indirect illumination component. The right side of Fig. 2 shows the various output of each injection stage.

3.1. *Direct illumination for translucent objects from area lights*

We first distribute a set of TPDS on the surface of a translucent object. For each Poisson disk sample, we perform a 'gathering' operation in which the transmitted flux from each refracted light ray from the area lights is computed. We create a ESLPV voxel structure of 32^3 resolution encompassing the AABB of the translucent objects. The light intensities are stored according to the refracted light directions. The transmitted flux entering the translucent medium from each refracted light ray is converted to its SH representation of a clamped cosine lobe and is accumulated into the voxels corresponding to their locations. We describe the transmitted flux at point x_i from a light ray emitted at a virtual point light (VPL) (from uniformly divided patch from the area light) location x_{light} from direction $\vec{\omega}$ as $L_t(x_i, \vec{\omega})$ in Equation (1).

$$L_t(x_i, \vec{\omega}) = \frac{T(\vec{\omega}, \vec{\omega}')(\vec{N} \cdot \vec{L}) * (\vec{N}_{light} \cdot -\vec{L}) * A * I_{intensity}}{|x_i - x_{light}|^2} \quad (1)$$

We use x_{light} as the representative point for each uniformly divided patch on the area light from uniform sampling. $T(\vec{\omega}, \vec{\omega}')$ refers to the Fresnel term for describing the fraction of light energy transmitted into the translucent material after entering from direction ω_k and refracted to direction $\vec{\omega_{k'}}$. A represents the area of a uniform patch on the area light defined by the VPL from uniform point sampling. A is used as part of the form factor computation between area to point contribution. $\vec{N}$ refers to the normal of the TPDS, and $\vec{L}$ refers to the normalized ray direction from the TPDS to the light. $I_{intensity}$ refers to the light intensity from the VPL. Equation 1 can be converted to its 2nd order Spherical Harmonics (SH) representation as in Equation (2) where the injected flux for each Poisson disk sample is represented in the SH coefficients, $c_{lm}^{(\vec{\omega}')}$, of a clamped cosine-lobe oriented at the refraction vector, $\vec{\omega}'$. y_{lm} refers to the SH basis functions. Fig. 3 describes how the refracted light energy is injected into the ESLPV voxels.

$$L_{t,lm}(x_i, \vec{\omega}) = \frac{[\sum_{l=0}^{\infty} \sum_{m=-l}^{l} c_{lm}^{(\vec{\omega}')} y_{lm}]}{\pi} * L_t(x_i, \vec{\omega}) \quad (2)$$

The SH coefficients in each ESLPV voxel are accumulated by summation and then inversely scaled by the total number of TPDS located in them. Once the light intensities are injected into the ESLPV voxels, we can perform the light propagation as usual, which creates the local information required for rendering the translucent appearance.

3.2. *Indirect illumination from area lights*

3.2.1. *Indirect illumination from direct area light sources*

As the usual LPV scheme does not handle area lights, our proposed solution generates a set of Poisson disk distributed samples [32] on the 3D scene we are rendering. The illumination for each Poisson disk sample is computed by performing a 'gathering' operation to the light source. We only gather contribution from rays that pass the visibility test. We compute the reflected flux, L_{VPL}, for each diffuse Poisson disk sample, x_i, which is a summation of the contribution of each individual

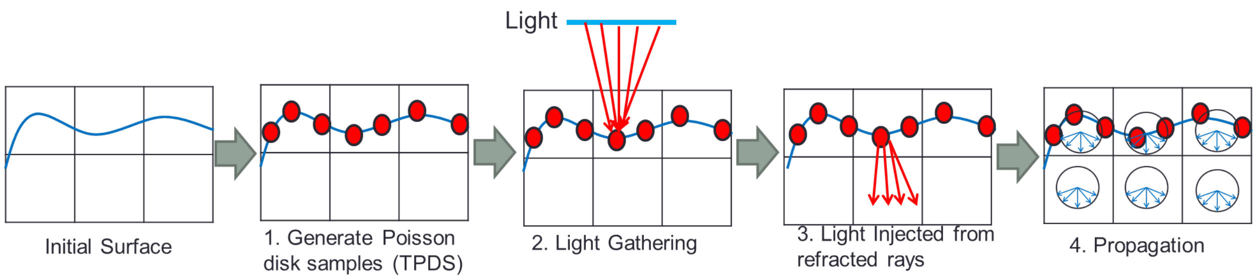


Fig. 3. For translucent materials, light is injected into the ESLPV voxels on the surface. The transmitted light intensity is stored in the voxels and propagated to render the local illumination of the translucent material.

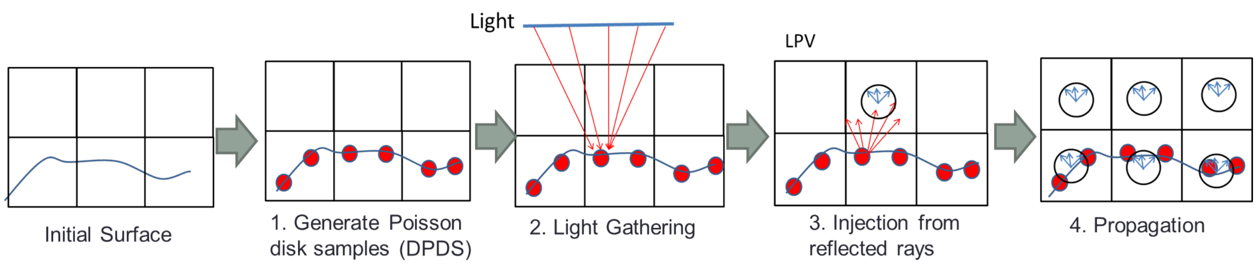


Fig. 4. For diffuse materials, the reflected flux is stored into the LPV and propagated to render the indirect illumination from the diffuse surfaces.

light ray to the sample point (refer to (3b)):

$$F_s = (\vec{N} \cdot \vec{L_k}) * (\vec{N}_{light} \cdot -\vec{L}_k), \tag{3a}$$

$$L_{VPL}(x_i, \vec{\omega}) = \frac{1}{\pi} \sum_{1}^{K} \frac{F_s * A * I_{intensity} * \rho}{|x_i - x_{k\_light}|^2} \tag{3b}$$

where $\vec{N}$ refers to the normal of Poisson disk sample x_i. $\vec{L_k}$ refers to the vector from x_i to a VPL, x_{k_light} on the light. $\vec{N}_{light}$ refers to the normal direction of the light. F_s refers to the foreshortening factor. $I_{intensity}$ refers to the intensity at x_{k_light} with area A from uniform sampling. A is used for area to point form factor computation. ρ refers to the diffuse coefficients of the Poisson disk sample.

The reflected flux in each Poisson disk sample is then deposited into the respective voxel at one unit normal distance away from the Poisson disk sample in their Spherical Harmonics (SH) representation. Following Kaplanyan et al. [28], we can first use a clamped cosine lobe oriented in the z-axis, represented by zonal harmonics, rotated to the direction $\vec{N}$. The flux distribution when converted to SH Coefficients is represented in Equation (4). As the accumulated reflected flux is defined as the integral of each individual light ray over a hemisphere, a normalization factor of $\frac{1}{\pi}$ is required to conserve the energy.

$$I(x_i, \vec{\omega}) = \frac{L_{VPL}(x_i, \vec{\omega})}{\pi} * \sum_{l=0}^{\infty} \sum_{m=-l}^{l} c_{lm}^{(\vec{N})} y_{lm} \tag{4}$$

In the LPV implementation, we are only required to store the four SH coefficients (two bands) multiplied by $\frac{L_{VPL}(x_i,\vec{\omega})}{\pi}$ into each voxel. Each color channel (R,G,B) is stored separately as four float values of SH coefficients each. Once we have obtained the SH coefficients by a cosine lobe orientated in direction $\vec{N}$, we can scale these coefficients by the irradiance of the Poisson disk sample. For Poisson disk samples that lie in the same voxels, the intensities can be accumulated in the same voxel.

The remaining process follows the standard LPV [28] pipeline. Propagation is performed to distribute the light intensity throughout the LPV. The final result is an approximation to the indirect illumination distributed by diffuse surfaces. We refer to Fig. 4 for the light injection process from area lights.

3.2.2. *Indirect illumination from translucent materials*

Scattered light from the translucent material can be represented by a sparse set of planar lights, with a point location allocated to its center. We downsample the set of TPDS that were generated earlier in Section 3.1 to get a reduced set of samples known as TPLs. The main intention for downsampling is to ensure that we are able to reduce the total number of samples for representing the translucent object. This reduces computation time significantly. For our downsampling algorithm, we reduce the 32^3 ESLPV to a 8^3 representation and temporarily deposit each TPDS into their corresponding locations in the 8^3 voxel volume. We then iterate each voxel of the 8^3 voxel volume and find the TPDS that is nearest to the center of each of the voxels in the 8^3 volume. Since the original Poisson disk sampling approach by Bowers et al. [32] distributes samples according to some voxel resolution, our downsampling approach would still allow us to reasonably obtain samples that are evenly distributed. The nearest TPDS in each voxel is chosen as the TPL. This chosen sample maintains its location on the surface of the object, unlike interpolation methods which might provide a non-surface sample. TPLs are represented as virtual planar light sources with an area allocated to them. We show this representation in Fig. 5. This representation corresponds to how voxels are represented as planar area lights in the SSLPV [21]. The output radiance of each TPL can be computed by Equation (5). A_{voxel} is the area of a voxel face of the ESLPV. There is no outgoing Fresnel term as it is incorporated when DPDS 'gather' illumination in the irradiance term from the TPL (6b).

$$I_{TPL}(x_{TPL}, \vec{\omega}_o = \vec{n}) = \frac{2}{A_{voxel}} \left(\sqrt{\frac{1}{4\pi}} c_{00}^{(\vec{n})} + \frac{1}{2} \sqrt{\frac{3}{4\pi}} c_{10}^{(\vec{n})} \right) \quad (5)$$

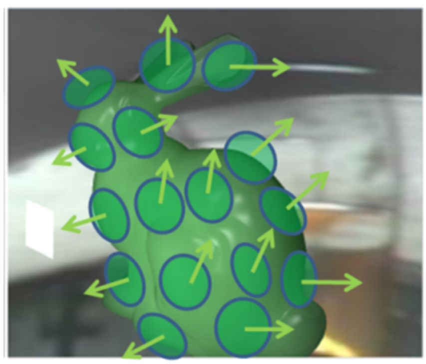


Fig. 5. The translucent object is represented with translucent planar lights. Each with an area, light intensity and normal orientation assigned to it.

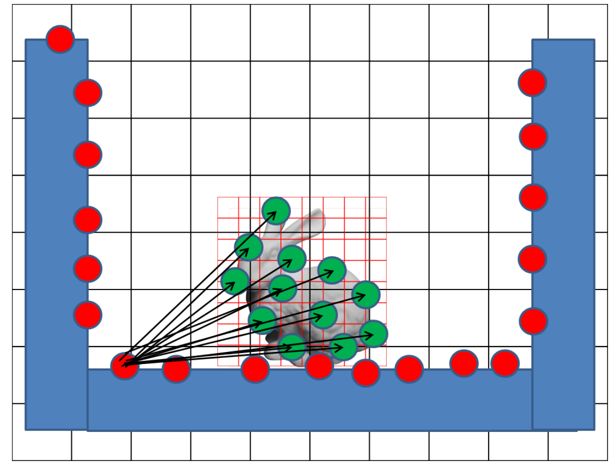


Fig. 6. Injecting light intensities into LPV with TPLs using a light gathering operation. DPDS are represented in red circles while TPLs are represented in green circles.

Once the intensity of each TPL is computed, each of the DPDS will perform a 'gathering' operation on the TPLs (refer to Fig. 6). DPDS are represented in red circles while TPLs are represented in green circles. The reflected flux contribution from each TPLs is then added to the original illumination contribution $L_{VPL}(x_i, \vec{\omega})$ from area lights in Section 3.2. This contribution is then stored into the respective voxel location corresponding to each of the DPDS' location. The reflected flux contributed by K number of TPLs is computed in Equation (6b), where K is the number of TPLs visible from x_i. $I_{TPL}(x_{TPL}, \vec{\omega}_o = \vec{n})$ is the intensity of the TPL computed by (5). $T(\vec{\omega}, \vec{\omega}')$ is the Fresnel transmission term describing the percentage of light energy leaving the translucent medium from direction $\vec{\omega}'$ to $\vec{\omega}$. F_s is the foreshortening factor as described in (3a). ρ is the albedo of the material at x_i. A_{TPL} refers to the surface area represented by each TPL. This is computed by dividing the total surface area of the translucent object with the total number of TPLs on the object.

$$L_{TPL\_Single}(x_i, \vec{\omega}) = \frac{F_s A_{TPL} T(\vec{\omega}, \vec{\omega}') I_{TPL}(x_{TPL}, \vec{\omega}' = \vec{n})}{|x_i - x_{TPL}|^2} \quad (6a)$$

$$L_{TPL}(x_i, \vec{\omega}) = \frac{\rho}{\pi} \sum_{1}^{K} L_{K\_TPL\_Single} \quad (6b)$$

4. Implementation

4.1. Poisson disk samples generation

We implemented Bowers et al. [32] Poisson disk sampling method as it properly distributes samples across a regular grid structure based on Euclidean distance. This ensures that each sample fulfills a minimum separation distance from its neighbouring samples. Although a GPU implementation of their algorithm has been created [35], we found that it is unnecessary to re-generate samples regularly. Due to the sparse number of samples we used, re-generating Poisson-disk samples per frame would also lead to temporal flickering. A new set of Poisson disk samples would only be required when a new diffuse surface or translucent object is added to the scene. In the case where the object in the scene is deformed, barycentric coordinates of the Poisson disk samples can be used to adapt to geometric changes.

In our work, we only control the maximum number of Poisson disk samples in each voxel. Under most conditions, each voxel would rarely contain the same number of samples, due to the difference in available surface area in each of the voxels. In our experiments, we find the maximum number of 5 Poisson disk samples to be sufficient for the LPV and the ESLPV. However, the maximum number of samples per voxel should be increased if there are thin planar surfaces. This is because Poisson disk samples may be distributed with higher probabilities on one side of the plane when sample size is low. In most cases, both voxel structures rarely contain the maximum number of samples that we specify. We show our experiment results with different number of TPDS for the ESLPV in Fig. 7. We show that the differences are unnoticeable between using 5 to 15 TPDS (Fig. 7a, Fig. 7b, Fig. 7c) when compared to 40 TPDS per ESLPV voxel for this scene. The normalized per pixel error seen in Fig. 7a is mainly within 5% of the reference image. Only significant errors are found in the ears of the bunny. The colored error chart is provided in Fig. 7e. For the LPV, 5 samples per voxel are sufficient to generate indirect illumination. We compare the varying maximum number of Poisson disk samples in each LPV voxel and their diffuse indirect illumination results in Fig. 8. There are little differences between using 5 to 15 samples as the errors are within 5% differences of a reference image using 40 samples per voxel. Due to the low frequency nature of the diffuse indirect illumination, the rendering differences are unnoticeable when using 10 or more samples as lighting differences in voxels are smoothed out during the propagation process.

4.2. Gathering operations on Poisson disk samples

As we need to generate multiple visibility rays from our Poisson disk samples to VPLs/TPLs, we utilize NVIDIA's CUDA and OptiX Prime [36] for our ray creation and ray tracing process. Firstly, a ray is created from every (diffuse and translucent) Poisson disk sample to a VPL on the area light source. The visibility tests are performed using OptiX Prime.

For every visible ray from a VPL to the TPDS, the transmitted flux is computed and compressed into its SH basis based on its refraction direction. $L_t(x_i, \vec{\omega})$ is projected into the SH coefficients of a clamped cosine lobe oriented at refracted direction $\vec{\omega}'$ in Equation (2). $c_{lm}^{(\vec{\omega}')}$ refers to the SH coefficients of the clamped cosine lobe based on the refracted ray direction $\vec{\omega}'$. Using CUDA's SHFL functions, the illumination can be gathered quickly if the samples are in multiples of warp sizes (16,32). Warps are units of threads which CUDA is able to execute in parallel. The gathered transmitted flux for each TPDS at position, x_i, is computed by accumulating the contribution of every visible ray in Equation (7). We only need to inject the values of four SH coefficients for each color channel, as described by $L^v_{t,lm}(x_i, \vec{\omega})$ in Equation (8), into the relevant voxel location of the ESLPV.

$$L(x_i, \vec{\omega}) = \sum_{1}^{K} L^k_{t,lm}(x_i, \vec{\omega}), \tag{7}$$

$$\begin{aligned} L^v_{t,lm}(x_i, \vec{\omega}) =& \text{float4}(\frac{c_{0,0} * L_t(x_i, \vec{\omega})}{\pi}, \frac{c_{1,-1}v * L_t(x_i, \vec{\omega})}{\pi}, \\ & \frac{c_{1,0} * L_t(x_i, \vec{\omega})}{\pi}, \frac{c_{1,1} * L_t(x_i, \vec{\omega})}{\pi}) \end{aligned} \tag{8}$$

Next, the TPDS is downsampled to TPLs, which are usually less than 256. All DPDS would then gather illumination in Equation (6b) from TPLs after the TPLs have been computed with the results from the ESLPV using Equation (5). Each TPL now represents an area light source with radiance and normals.

4.3. Indirect illumination voxels

Reflected flux from all diffuse Poisson disk samples, after 'gathering' from VPLs and TPLs, are injected into the LPV. The SH coefficients of the reflected flux are computed by multiplying the reflected flux of the Poisson disk sample and the clamped cosine SH lobe of the normal vector of the Poisson disk sample. The SH coefficients of the reflected flux from the Poisson disk samples are injected into the LPV at an offset of one unit voxel away in the normal direction. We keep a counter using a 3D texture and GLSL's imageAtomicAdd function to keep track of the number of samples inside each voxel. Ultimately, the intensities stored in the voxels should be normalized by the number of samples that receive light in them. This normalization is not required in the original ESLPV and LPV as the normalization has already been done by dividing the total light energy by the number of texels in the RSM. However, in our case we are unable to use a RSM to represent an area light.

4.4. Flux Propagation

The basic idea of flux propagation in the LPV [28] is to accumulate the contribution of flux from the six neighbouring voxels of the target voxel. To compute the contribution of flux from a neighbouring voxel, we need to determine the amount of flux from the center of the neighbouring voxel that reach each of the voxel faces, except for the bordering face connecting the two voxels, in the destination cell. Hence for one source cell s, its flux contribution Φ to its neighbouring cell d on face f is described by:

$$\Phi_{f,d\leftarrow s} = \int_{\Delta\omega_{f,s}} I_s(\omega) d\vec{\omega}, \tag{9}$$

where $\Delta\omega(f, s)$ is the solid angle subtended by center of the source cell to the face f of the neighbouring cell.

The flux computed in Equation (9) is projected as a new VPL in the destination voxel. The point light representation of this flux contribution can be converted into the form of a clamped cosine lobe centered towards face f as shown in equation 10.

$$I_{f,d\leftarrow s} = \frac{\Phi_{f,d\leftarrow s}}{\pi} max(0, n_f \cdot \vec{\omega}), \tag{10}$$

where n_f is the face normal of the destination voxel face.

The flux contributions to every face f of the destination cell, from every neighbour voxel s are accumulated to the destination voxel's VPL. The flux transfer in Equation (9) can be expressed in its SH form as shown in Equation (13) since the radiance in the source cell is already in its SH form after the injection stage.

$$\Phi_{f,d\leftarrow s} = \int_{\Delta\omega_{f,s}} I_s(\omega) d\vec{\omega} \tag{11}$$

$$= \sum_{l,m} c_s \int_{f,d\leftarrow s} y_{lm}(\vec{\omega}) d\vec{\omega} \tag{12}$$

$$= v^T_{f,d\leftarrow s} \cdot c_s \tag{13}$$

$$v^T_{f,d\leftarrow s} = \int_{f,d\leftarrow s} y_{lm}(\vec{\omega}), \tag{14}$$

c_s is the SH coefficient vector of the radiance stored in the source voxel (initially supplied from the injection stage) and $v^T_{f,d\leftarrow s}$ is the transfer vector which maps the source radiance (in SH) to the destination flux for a particular target voxel face, f. To compute the new SH Coefficients, $c_{f,d\leftarrow s}$, of the flux contribution from the source to destination voxels, we can simply use Equation (16) after substituting Equation (10) into Equation (15).

$$c_{f,d\leftarrow s} = \begin{bmatrix} \int_{4\pi} I_{f,d\leftarrow s} y_{00}(\vec{\omega}) d\vec{\omega} \\ \int_{4\pi} I_{f,d\leftarrow s} y_{1-1}(\vec{\omega}) d\vec{\omega} \\ \int_{4\pi} I_{f,d\leftarrow s} y_{10}(\vec{\omega}) d\vec{\omega} \\ \int_{4\pi} I_{f,d\leftarrow s} y_{11}(\vec{\omega}) d\vec{\omega} \end{bmatrix} \tag{15}$$

$$= \frac{1}{\pi} \begin{bmatrix} \int_{2\pi} n_f \cdot \vec{\omega} y_{00}(\vec{\omega}) d\vec{\omega} \\ \int_{2\pi} n_f \cdot \vec{\omega} y_{1-1}(\vec{\omega}) d\vec{\omega} \\ \int_{2\pi} n_f \cdot \vec{\omega} y_{10}(\vec{\omega}) d\vec{\omega} \\ \int_{2\pi} n_f \cdot \vec{\omega} y_{11}(\vec{\omega}) d\vec{\omega} \end{bmatrix} * v^T_{f,d\leftarrow s} \cdot c_s \tag{16}$$

Fortunately, the SH basis functions, $y_{lm}(\vec{\omega})$, can be easily precomputed and stored for every face direction (+X, -X, +Y, -Y, +Z, -Z). The final dot product $v^T_{f,d\leftarrow s} \cdot c_s$ in Equation (16) can be computed with the dot product of SH coefficient vector c_s and the basis function y_{lm}, while the multiplication with the front matrix $\int_{2\pi} n_f \cdot \vec{\omega} y_{lm}(\vec{\omega}) d\vec{\omega}$ simply projects the flux back into SH basis with a scaling factor dependent on the solid angle $\Delta\omega_{f,s}$.

The blocking potential, $B(\omega)$, retrieved from geometric normals in Doghramachi's LPV [31] can be further multiplied with $c_{f,d\leftarrow s}$ to limit the amount of flux being transferred to the destination voxel due to possible occlusion. Blocking geometry can also simulate multiple light bounce effects. The reflected light from multiple bounces possesses the same directional distribution, $B(\omega)$, as the geometric blockers. Hence the final light intensity in each voxel is equivalent to the sum of flux received from its neighbours and flux that is reflected back from its neighbours.

5. Results and discussion

We show the rendering results of four scenes: a translucent water glass, a translucent Buddha model and a plane, a translucent bunny in a colored Cornell box, a translucent dragon in the Sponza scene in Figs. 1a, 9a, 10a, 12, respectively, in 1024x768 pixel resolution. We tabulate the computation timings in Table 1. The rendering was performed on an Intel i5 3.40GHz CPU with an NVIDIA GeForce GTX 980 GPU. The direct illumination for diffuse surfaces in our results are rendered with a screenspace multi-resolution technique [34], with 64 samples per fragment. The direct illumination provides the soft shadows effects. The translucent materials in our reference images (generated by Mitsuba [37]) are rendered with Photon Beam Diffusion [6]. Due to the sparse distribution of the Poisson disk samples, light 'gathering' operations are considered to be cheap operations that can be computed in tens of milliseconds. Furthermore, these 'gathering' operations only need to be performed when there is a change in object positions, material properties or light positions. Figure 12 being the most computation intensive result, uses approximately 61MB of pre-allocated memory on the GPU and RAM for its ray tracing infrastructure and 1MB for storing its DPDS, TPDS, and TPLs. The LPV and ESLPV at its 32^3 structure uses approximately 20MB of pre-allocated memory.

In Fig. 1a, we placed an area light inside a water glass so that it would be easier to only see the indirect illumination produced by the translucent materials as no direct illumination would be able to penetrate the water glass. Our result produced a reddish color bleed as observed on the floor outside the water glass. The reddish color is also observed in the reference image in Fig. 1b. In Fig. 9, we illuminate a Buddha model floating on top of a planar surface. Fig. 9a shows the soft shadow produced by the Buddha model. The shadow is not completely dark and has a yellowish faint partially illuminated by light from the translucent material. These effects are more evident in the soft shadow region of the shadows. A reference image is provided in Fig. 9b. In Fig. 10a, we can see that there is faint coloration of reddish and greenish in the shadows of the bunny. This coloration is created by the indirect illumination from both the colored walls of the Cornell box and the scattered light from the bunny. We show that the rendered reference image with the Mitsuba renderer's path tracer [37] in Fig. 10b and the coloration in their shadows are similar to ours. In Fig. 12a, we specifically render the indirect illumination only. It can be seen that scattered green light is present around the diffuse surfaces near the dragon. The scattered green light is also present in the soft shadows areas formed by the pillars or the dragon in Figs. 12b, 12c. In Fig. 11, a translucent bunny is placed in the Sibenik scene. The translucent material provides a reddish glow to its surrounding environment. The red coloration in the shadows exhibits this effect.

With reference to Table 1, 'Direct Illum.' refers to the time used for rendering direct illumination (excluding translucent material rendering) using a multi-resolution screenspace approach [34]. 'Poisson Gathering' refers to the time used in 'gathering' illumination from Poisson disk samples generated on diffuse and translucent surfaces after being lit by VPLs. 'TPL Gathering' refers to the time needed for all diffuse Poisson disk samples to gather light from TPLs. 'ESLPV' represents the time needed to render the illumination on the translucent object using our ESLPV method. 'LPV' represents the time required for the LPV to render indirect illumination on the entire scene. We can observe that direct illumination of translucent materials using area lights can be quickly computed and rendered in less than 10 ms. Indirect illumination for the entire scene can be rendered in less than 20 ms for most of our results. 'Total time' refers to the time required for rendering a single frame with both direct and indirect illuminations. The total time for rendering appears high due to the direct illumination component. However, our indirect illumination technique is independent of the direct illumination technique used. The columns 'DPDS', 'TPDS' and 'TPL' describe the number of samples required for diffuse Poisson disk samples, translucent Poisson disk samples and Translucent Planar Lights, respectively. One advantage of voxel-based techniques is that the timings for ESLPV and LPV do not scale up with the scene complexity, which allows our indirect illumination for translucent materials to be rendered at fast speed.

In Fig. 13, we compare the differences between Poisson Disk samples and random sampling for the rendered image in Fig.11. In Fig. 13a and 13b, the Poisson disk samples is able to illuminate the top left of the image correctly when compared to random sampling in Fig. 13c, 13d. Fig. 13h shows the planar light location used in the scene. Although both sampling methods are using the same number of samples (12k), Poisson disk sampling provides a better coverage on areas especially the illuminated side of the pillar. A zoomed in image is provided in Fig. 13e and Fig. 13f. The results in this figure shows the importance of using Poisson disk samples when sample density are sparse. The lack of coverage of samples in areas overlapped by voxels will create a lack of irradiance. In Fig. 13g, the LPV voxels are visualized. In the LPV implementation, voxels are clamped to the camera position, and its size is determined by furthest visible triangle in the scene.

We do note that there are some differences compared to the reference images in some areas of the images. In Fig. 10, we note that our result (Fig. 10a) has a different ceiling color compared to the reference image. This is because the LPV is not a physically-based algorithm and it uses a very abstract representation of geometry and reflectance property to distribute indirect illumination. Hence, we cannot expect the same quality of results as that in radiosity, photon mapping or path-tracing methods when rendering multi-bounce illumination. However, we can still expect color-bleeding effects to be rendered. We are unable to simulate high frequency effects for translucent materials with weak scattering properties due to the highly compressed nature of our spherical harmonics representation. However, this is compensated through its low-memory consumption and fast rendering speed. While Bowers et al. [32] parallel Poisson-disk sampling uses a crude approximation for geodesic distance between each sample point, they are much closer to a minimum Euclidean spacing. This sampling method is likely to cause problems in some geometry such as a thin plane. It is highly possible for one side of the plane geometry to receive more samples than the other side of the plane geometry. Optimally, what we should do is to allocate samples with a minimum geodesic distance such as Fu et al.'s [38]. Alternatively, photon relaxation techniques [39] can be used to adaptively assign local blue noise samples to areas that are more significantly important, such as those with high flux gradient differences.

6. Conclusion and Future Work

We have presented an efficient and low cost solution for illumination of translucent materials from area lights. This is first done by generating a set of Poisson disk samples over the translucent materials and performing an illumination 'gathering' operation over them. The Poisson disk samples are then injected into the ESLPV [22], which distributes the light intensities through the translucent materials. Similarly, Poisson disk samples are distributed around diffuse surfaces in the LPV for gathering direct illumination from the area lights.

In order to further illuminate the scene using the translucent objects, each of the Poisson disk samples in the LPV will gather illumination from a downsampled set of translucent planar lights (TPLs) on the translucent objects. These samples are then injected into the LPV. The LPV distributes the light, simulating indirect illumination with contributions from the diffuse and translucent objects. Our method efficiently distributes indirect illumination with very little computation overheads.

Using a Poisson disk sampling approach, we ensure that translucent materials and their indirect illuminations can be rendered with good quality under area lighting. This method achieves interactive rendering while remaining precomputationless and maintaining low storage costs.

The diffuse Poisson disk samples (DPDS) in the scene increase with scene complexity, hence leading to a overall increase in time for 'gathering' illumination from TPLs. A multi-resolution approach should be designed to select an appropriate set of samples. Other optimizations such as Debevec's regular light probe sampling [40] could be more efficient in choosing our TPLs from our TPDS. However, it was designed for sampling a 2D environment map but it can well be extended to simulate importance sampling for clustering of TPLs. Alternatively, methods such as hierarchical photon mapping [41] can be used for clustering TPLs on light source.

Acknowledgments

This research is partially supported by the National Research Foundation, Prime Minister's Office, Singapore under its International Research Centres in Singapore Funding Initiative. The 3D models used in our examples are obtained from: http://graphics.cs.williams.edu/data.

References

[1] Pharr, M, Hanrahan, P. Monte Carlo evaluation of non-linear scattering equations for subsurface reflection. In: Proceedings of the 27th Annual Conference on Computer Graphics and Interactive Techniques. SIGGRAPH '00; New York, NY, USA: ACM Press/Addison-Wesley Publishing Co. ISBN 1-58113-208-5; 2000, p. 75–84. doi:10.1145/344779.344824.

[2] Jensen, HW, Legakis, J, Dorsey, J. Rendering of wet materials. In: Proceedings of the 10th Eurographics Conference on Rendering. EGWR'99; Aire-la-Ville, Switzerland, Switzerland: Eurographics Association. ISBN 3-211-83382-X; 1999, p. 273–282. doi:10.2312/EGWR/EGWR99/273-282.

[3] Donner, C, Jensen, HW. Rendering translucent materials using photon diffusion. In: ACM SIGGRAPH 2008 Classes. SIGGRAPH '08; New York, NY, USA: ACM; 2008, p. 4:1–4:9. doi:10.1145/1401132.1401138.

[4] Jarosz, W, Jensen, HW, Donner, C. Advanced global illumination using photon mapping. In: ACM SIGGRAPH 2008 Classes. SIGGRAPH '08; New York, NY, USA: ACM; 2008, p. 2:1–2:112. doi:10.1145/1401132.1401136.

[5] Sheng, Y, Shi, Y, Wang, L, Narasimhan, SG. A practical analytic model for the radiosity of translucent scenes. In: Proceedings of the ACM SIGGRAPH Symposium on Interactive 3D Graphics and Games. I3D '13; ACM. ISBN 978-1-4503-1956-0; 2013, p. 63–70.

[6] Habel, R, Christensen, PH, Jarosz, W. Photon beam diffusion: A hybrid monte carlo method for subsurface scattering. Computer Graphics Forum (Proceedings of EGSR 2013) 2013;32(4).

[7] Lensch, HPA, Goesele, M, Bekaert, P, Kautz, J, Magnor, MA, Lang, J, et al. Interactive rendering of translucent objects. In: Proceedings of the 10th Pacific Conference on Computer Graphics and Applications. PG '02; IEEE Computer Society. ISBN 0-7695-1784-6; 2002, p. 214–.

[8] Mertens, T, Kautz, J, Bekaert, P, Van Reeth, F, Seidel, HP. Efficient rendering of local subsurface scattering. In: Proceedings of the 11th Pacific Conference on Computer Graphics and Applications. PG '03; IEEE Computer Society. ISBN 0-7695-2028-6; 2003, p. 51–58.

[9] Carr, NA, Hall, JD, Hart, JC. GPU algorithms for radiosity and subsurface scattering. In: Proceedings of the ACM SIGGRAPH/EUROGRAPHICS conference on Graphics hardware. HWWS '03; Eurographics Association. ISBN 1-58113-739-7; 2003, p. 51–59.

[10] Wang, Y, Wang, J, Holzschuch, N, Subr, K, Yong, JH, Guo, B. Real-time Rendering of Heterogeneous Translucent Objects with Arbitrary Shapes. vol. 29. Wiley; 2010, p. 497–506. doi:10.1111/j.1467-8659.2009.01619.x.

[11] Arbree, A, Walter, B, Bala, K. Heterogeneous subsurface scattering using the finite element method. vol. 17. Piscataway, NJ, USA: IEEE Educational Activities Department; 2011, p. 956–969. doi:10.1109/TVCG.2010.117.

[12] Borshukov, G, Lewis, JP. Realistic human face rendering for "the matrix reloaded". In: ACM SIGGRAPH 2003 Sketches &Amp; Applications. SIGGRAPH '03; New York, NY, USA: ACM; 2003, p. 1–1. URL: http://doi.acm.org/10.1145/965400.965470. doi:10.1145/965400.965470.

[13] Jimenez, J, Gutierrez, D. Faster rendering of human skin. In: In Proceedings of the CEIG. 2008, p. 21–28.

Table 1. Rendering statistics for 1024x768 images except for Fig. 11 which is rendered in 1280x960. Direct Illum refers to the time required to render direct illumination [34] only for diffuse surfaces. The remaining columns (Poisson Gathering, TPL Gathering, ESLPV, LPV) are timings for our work.

Figure	Triangles	Direct Illum. [34](ms)	Poisson Gathering (ms)	TPL Gathering (ms)	ESLPV (ms)	LPV (ms)	Total time (ms)/(FPS)	DPDS	TPDS	TPL
1a - Water glass	9,280	35	3	9	5	9	61 /16.4	3,711	17,077	179
9a - Buddha	110,346	77	3	5	2	10	97/10.1	3,680	4,913	236
10a - Bunny in Cornell Box	80,020	90	4	9	5	9	117/8.55	2,603	8,438	177
11 - Bunny in Sibenik	155,046	66	5	21	5	9	106/9.43	12,904	6,425	179
12b - Dragon in Sponza	238,076	80	6	24	5	9	124/8.06	21,366	8,495	84

[14] d'Eon, E, Luebke, D, Enderton, E. Efficient rendering of human skin. In: Proceedings of the 18th Eurographics conference on Rendering Techniques. EGSR'07; Eurographics Association. ISBN 978-3-905673-52-4; 2007, p. 147–157.

[15] Jimenez, J, Sundstedt, V, Gutierrez, D. Screen-space perceptual rendering of human skin. ACM Trans Appl Percept 2009;6(4):23:1–23:15. URL: http://doi.acm.org/10.1145/1609967.1609970. doi:10.1145/1609967.1609970.

[16] Jimenez, J, Whelan, D, Sundstedt, V, Gutierrez, D. Real-time realistic skin translucency. Computer Graphics and Applications, IEEE 2010;30(4):32–41.

[17] Echevarria, JI, Munoz, A, Seron, FJ, Gutierrez, D. Screen-space rendering of translucent objects. In: XX Congreso Español de Informatica Grafica (CEIG 2010). Eurographics; 2010, p. 127–133.

[18] Munoz, A, Echevarria, JI, Seron, FJ, Gutierrez, D. Convolution-based simulation of homogeneous subsurface scattering. vol. 30. Blackwell Publishing Ltd; 2011, p. 2279–2287.

[19] Geist, R, Rasche, K, Westall, J, Schalkoff, R. Lattice-Boltzmann lighting. In: Proceedings of the Fifteenth Eurographics conference on Rendering Techniques. EGSR'04; Eurographics Association. ISBN 3-905673-12-6; 2004, p. 355–362.

[20] Bernabei, D, Hakke-Patil, A, Banterle, F, Di Benedetto, M, Ganovelli, F, Pattanaik, S, et al. A parallel architecture for interactively rendering scattering and refraction effects. Computer Graphics and Applications, IEEE 2012;32(2):34–43.

[21] Børlum, J, Christensen, BB, Kjeldsen, TK, Mikkelsen, PT, Noe, KO, Rimestad, J, et al. SSLPV: Subsurface light propagation volumes. In: Proceedings of the ACM SIGGRAPH Symposium on High Performance Graphics. HPG '11; ACM. ISBN 978-1-4503-0896-0; 2011, p. 7–14.

[22] Koa, MD, Johan, H. ESLPV: Enhanced subsurface light propagation volumes. The Visual Computer 2014;30(6-8):821–831. doi:10.1007/s00371-014-0952-3.

[23] Crassin, C, Neyret, F, Sainz, M, Green, S, Eisemann, E. Interactive indirect illumination using voxel cone tracing. Computer Graphics Forum (Proceedings of Pacific Graphics 2011) 2011;30(7).

[24] Sugihara, M, Rauwendaal, R, Salvi, M. Layered Reflective Shadow Maps for Voxel-based Indirect Illumination. In: Wald, I, Ragan-Kelley, J, editors. Eurographics/ ACM SIGGRAPH Symposium on High Performance Graphics. The Eurographics Association. ISBN 978-3-905674-60-6; 2014,.

[25] Dachsbacher, C, Stamminger, M. Reflective shadow maps. In: Proceedings of the 2005 Symposium on Interactive 3D graphics and games. I3D '05; ACM. ISBN 1-59593-013-2; 2005, p. 203–231.

[26] Crassin, C, Luebke, D, Mara, M, McGuire, M, Oster, B, Shirley, P, et al. CloudLight: A system for amortizing indirect lighting in real-time rendering. Journal of Computer Graphics Techniques (JCGT) 2015;4(4):1–27. URL: http://jcgt.org/published/0004/04/01/.

[27] Iwanicki, M, Sloan, PP. Precomputed lighting in call of duty: Infinite warfare. In: ACM SIGGRAPH 2017 Courses: Advances in real-time rendering in Games. SIGGRAPH '17; New York, NY, USA: ACM. ISBN 978-1-4503-5014-3; 2017,doi:10.1145/3084873.3096477.

[28] Kaplanyan, A, Dachsbacher, C. Cascaded light propagation volumes for real-time indirect illumination. In: Proceedings of the 2010 ACM SIGGRAPH Symposium on Interactive 3D Graphics and Games. I3D '10; ACM. ISBN 978-1-60558-939-8; 2010, p. 99–107.

[29] Hedman, P, Karras, T, Lehtinen, J. Sequential Monte Carlo instant radiosity. In: Proceedings of the 20th ACM SIGGRAPH Symposium on Interactive 3D Graphics and Games. I3D '16; New York, NY, USA: ACM. ISBN 978-1-4503-4043-4; 2016, p. 121–128. doi:10.1145/2856400.2856406.

[30] Schwarz, M, Seidel, HP. Fast parallel surface and solid voxelization on gpus. ACM Trans Graph 2010;29(6):179:1–179:10.

[31] Doghramachi, H. GPU Pro 4: Advanced Rendering Techniques - Rasterized Voxel-Based Dynamic Global Illumination. Natick, MA, USA: A. K. Peters/CRC Press; 2013.

[32] Bowers, J, Wang, R, Wei, LY, Maletz, D. Parallel Poisson disk sampling with spectrum analysis on surfaces. ACM Trans Graph 2010;29(6):166:1–166:10. doi:10.1145/1882261.1866188.

[33] Subr, K, Singh, G, Jarosz, W. Fourier analysis of numerical integration in monte carlo rendering: Theory and practice. In: ACM SIGGRAPH Courses. New York, NY, USA: ACM; 2016,doi:10.1145/2897826.2927356.

[34] Koa, MD, Johan, H, Sourin, A. Interactive screenspace fragment rendering for direct illumination from area lights using gradient aware subdivision and radial basis function interpolation. Computers And Graphics 2017;64:37 – 50. doi:https://doi.org/10.1016/j.cag.2017.01.003.

[35] Ying, X, Xin, SQ, Sun, Q, He, Y. An intrinsic algorithm for parallel Poisson disk sampling on arbitrary surfaces. IEEE Transactions on Visualization and Computer Graphics 2013;19(9):1425–1437. doi:10.1109/TVCG.2013.63.

[36] Parker, SG, Bigler, J, Dietrich, A, Friedrich, H, Hoberock, J, Luebke, D, et al. Optix: a general purpose ray tracing engine. ACM Trans Graph 2010;29(4):66:1–66:13.

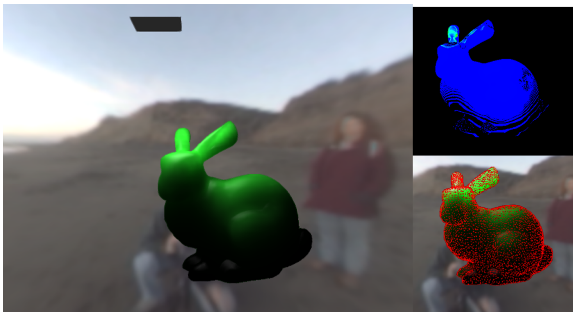
[37] Jakob, W. Mitsuba renderer. 2010. Http://www.mitsuba-renderer.org.

[38] Fu, Y, Zhou, B. Direct sampling on surfaces for high quality remeshing. In: Proceedings of the 2008 ACM Symposium on Solid and Physical Modeling. SPM '08; New York, NY, USA: ACM. ISBN 978-1-60558-106-4; 2008, p. 115–124. doi:10.1145/1364901.1364919.

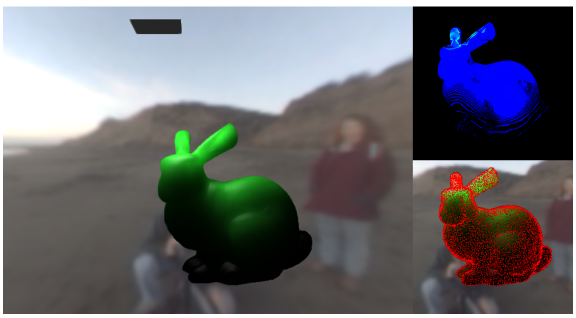
[39] Spencer, B, Jones, M. Photon parameterisation for robust relaxation constraints. Computer Graphics Forum 2013;32(2pt1):83–92. URL: http://dx.doi.org/10.1111/cgf.12028. doi:10.1111/cgf.12028.

[40] Debevec, P. A median cut algorithm for light probe sampling. In: ACM SIGGRAPH 2006 Courses. SIGGRAPH '06; New York, NY, USA: ACM. ISBN 1-59593-364-6; 2006,doi:10.1145/1185657.1185688.

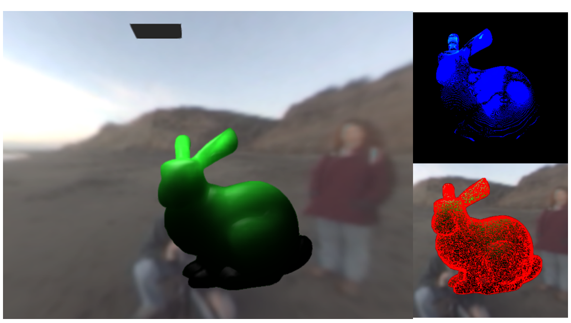
[41] Spencer, B, Jones, MW. Hierarchical photon mapping. IEEE Transactions on Visualization and Computer Graphics 2009;15(1):49–61. doi:10.1109/TVCG.2008.67.



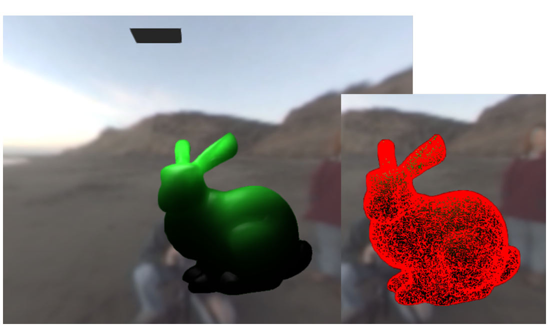
(a) Max. 5 samples per voxel



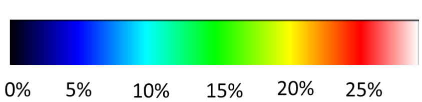
(b) Max. 10 samples per voxel



(c) Max. 15 samples per voxel

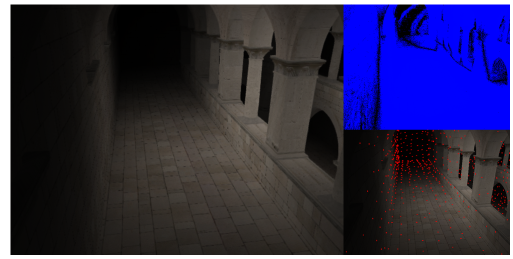


(d) Reference Max. 40 samples per voxel

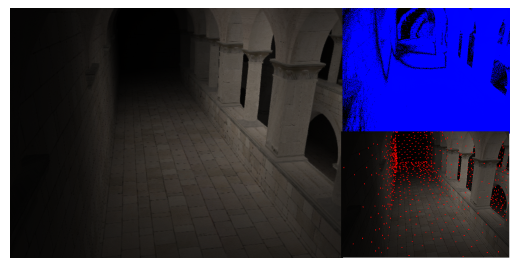


(e) Error Chart

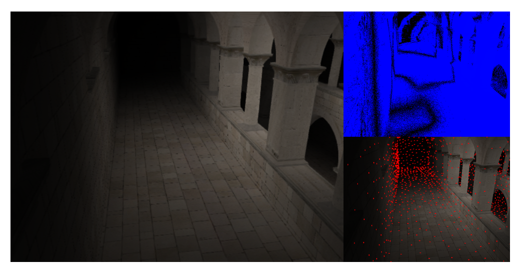
Fig. 7. Rendering of a Stanford Bunny object with a material property simulating jade. The Poisson disk parameters are tuned to maximum of 5, 10, 15 samples per voxel. The small inset image on the bottom right of each figure shows the Poisson disk samples distribution (represented by red dots). The top right image of each figure shows the normalized per pixel error representation with respect to the reference image in (d). (e) shows the color mapping with the percentage error.



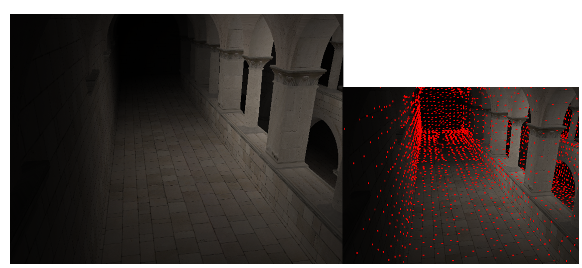
(a) Max. 5 samples per voxel



(b) Max. 10 samples per voxel

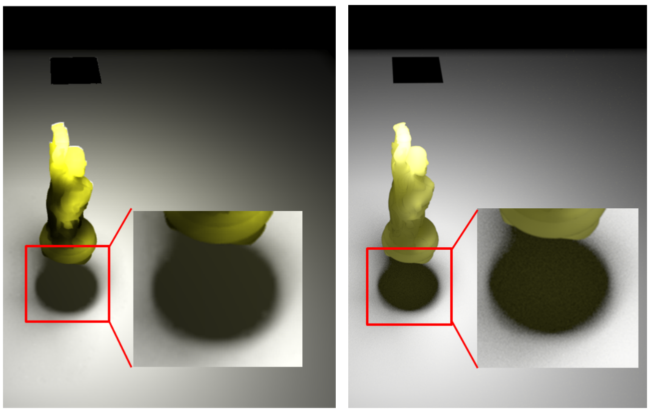


(c) Max. 15 samples per voxel



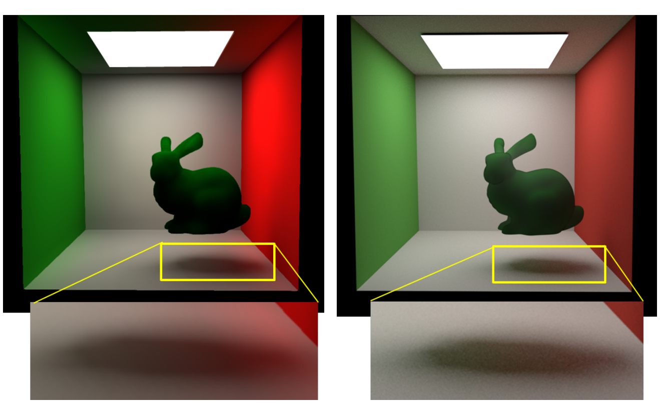
(d) Reference Max. 40 samples per voxel

Fig. 8. Rendering of indirect illumination of the Sponza corridor under area light illumination. The Poisson disk parameters are tuned to maximum of 5, 10, 15 samples per voxel. The small inset image on the bottom right of each figure shows the Poisson disk samples distribution (represented by red dots). The top right image of each figure shows the normalized per pixel error representation with respect to the reference image in (d). The error chart is provided in Fig.7e.



(a) Our result (b) Mitsuba Path Tracer Global Illumination

Fig. 9. Rendering of a Buddha object with a material property simulating an apple in (a) and (b). We show our rendering in (a) with both the direct and indirect illumination components, and the reference generated by Mitsuba Path Tracer (64 samples per pixel) in (b) with the Photon Beam Diffusion method [6]. Image (b) is rendered in 28.97 seconds.



(a) Our result (b) Mitsuba Path Tracer Global Illumination

Fig. 10. Rendering of a Stanford bunny with material properties simulating jade in a colored Cornell box. We show our rendering in (a) with both the direct and indirect illumination components, and the reference generated by Mitsuba Path Tracer (64 samples per pixel) and the Photon Beam Diffusion method [6] in (b). Image (b) is rendered in 54.21 seconds.

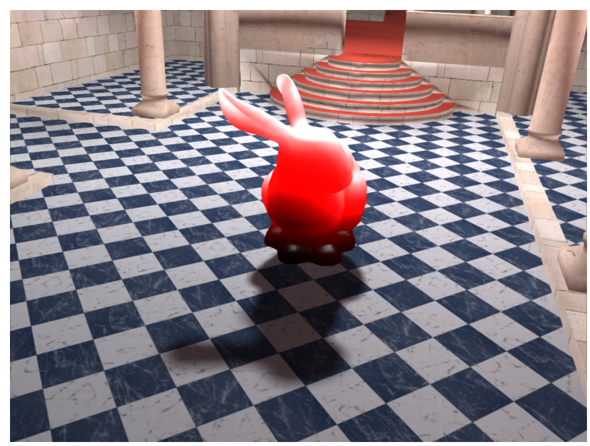
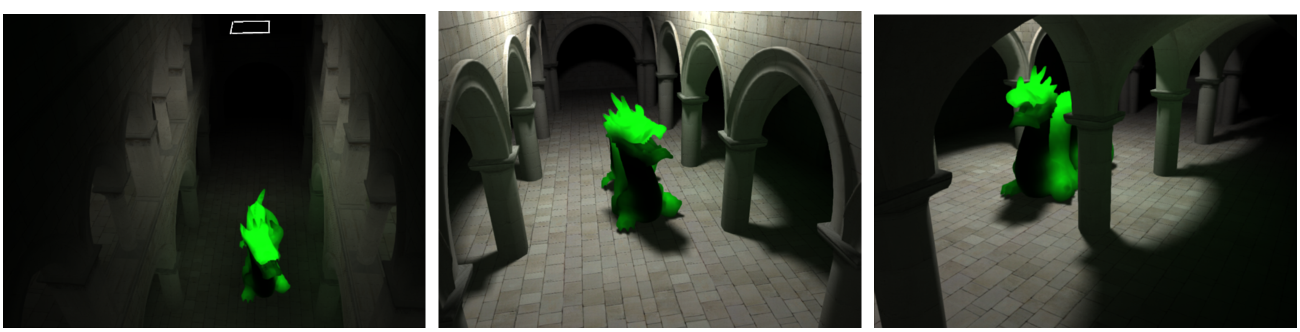
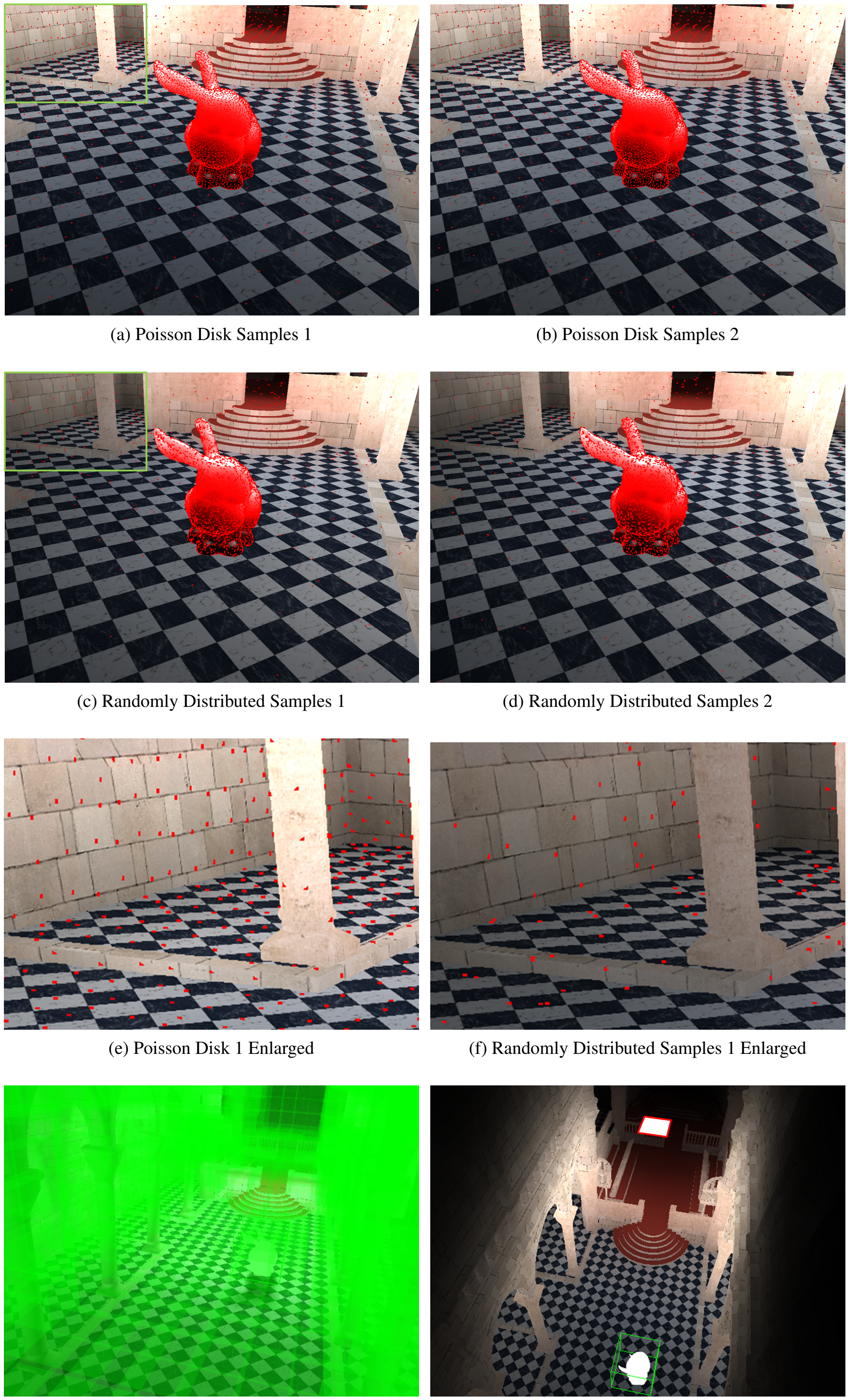


Fig. 11. Rendering of a Stanford bunny with red scattering materials in a Sibenik scene. The image is rendered with both direct and indirect illumination components in 1280x960 resolution.



(a) Indirect Illumination only (b) Dragon View 1 (c) Dragon View 2

Fig. 12. Using TPLs for injection produces a soft diffuse look from the green translucent Stanford dragon model. The scene is illuminated by a rectangular area light source from the top. Image (a) shows the indirect illumination. The walls and pillars of the lower level which are closer to the dragon appear much greener compared to the top level walls and pillars. The light source used in the scene is a small white rectangular area light. Images (b) and (c) show other rendered views with the direct and indirect illumination components. The soft shadow regions exhibit indirect scattered illumination from the translucent material.



(a) Poisson Disk Samples 1

(b) Poisson Disk Samples 2

(c) Randomly Distributed Samples 1

(d) Randomly Distributed Samples 2

(e) Poisson Disk 1 Enlarged

(f) Randomly Distributed Samples 1 Enlarged

(g) Voxel Structure

(h) Lighting Location

Fig. 13. Samples distribution of a diffuse Sibenik scene with a translucent red Stanford bunny as per Fig. 11. Images (a) and (b) shows the indirect illumination contributed by the Poisson disk samples. Images (c) and (d) shows the indirect illumination contributed by the Poisson disk samples. An enlarged top left inset of (a) and (c) is provided in images (e) and (f) respectively. Image (g) shows the LPV voxel structure of the scene. Image (h) shows the location of the light source (white square with red borders).