Combining Global and Local Virtual Lights for Detailed Glossy Illumination

Tomáš Davidovič*
Saarland University and DFKI

Jaroslav Křivánek
Charles University, Prague
Cornell University

Miloš Hašan
Harvard University

Philipp Slusallek
Saarland University and DFKI

Kavita Bala
Cornell University

Figure 1: Comparison of our approach with Virtual Spherical Lights (VSLs). *Left: VSLs fail to capture small local glossy reflections (time: 6 m 26 s). Right: our method (time: 4 m 59 s) computes these reflections more efficiently and accurately by using visibility approximations for the low-rank global light transport, and local lights for the high-rank localized light transport.*

Abstract

Accurately rendering glossy materials in design applications, where previewing and interactivity are important, remains a major challenge. While many fast global illumination solutions have been proposed, all of them work under limiting assumptions on the materials and lighting in the scene. In the presence of many glossy (directionally scattering) materials, fast solutions either fail or degenerate to inefficient, brute-force simulations of the underlying light transport. In particular, many-light algorithms are able to provide fast approximations by clamping elements of the light transport matrix, but they eliminate the part of the transport that contributes to accurate glossy appearance. In this paper we introduce a solution that separately solves for the global (low-rank, dense) and local (high-rank, sparse) illumination components. For the low-rank component we introduce visibility clustering and approximation, while for the high-rank component we introduce a *local light* technique to correct for the missing illumination. Compared to competing techniques we achieve superior gloss rendering in minutes, making our technique suitable for applications such as industrial design and architecture, where material appearance is critical.

CR Categories: K.6.1 [Computer Graphics]: Three-Dimensional Graphics and Realism—Color, shading, shadowing, and texture;

Keywords: global illumination, glossy interreflection, many lights

*e-mail: davidovic@cs.uni-saarland.de, krivanek@ksvi.mff.cuni.cz, milos.hasan@gmail.com, slusallek@cs.uni-saarland.de, kb@cs.cornell.edu

1 Introduction

Accurate rendering in the presence of glossy materials in design applications, where effective previews are important, remains a major challenge. Existing techniques trade off between performance and accuracy. For example, pure Monte Carlo solutions are perfectly accurate given enough samples, but very slow. On the other hand, interactive techniques achieve performance by limiting their support for materials by assuming diffuse or low-frequency material representations, sometimes extensible to perfectly specular [Wang et al. 2009], and/or requiring significant precomputation, as in PRT-based algorithms [Cheslack-Postava et al. 2008]. Instant radiosity-based algorithms are gaining popularity because of their simplicity and their ability to effectively leverage GPU performance. These formulations convert global illumination into the problem of rendering with many virtual point lights (VPLs). These approaches assume that any individual VPL does not significantly affect lighting. Thus, "spiky" lights (either because of BRDF or proximity) are eliminated through clamping their contribution to some user-specified maximum value. However, clamping can negatively impact accurate material perception [Křivánek et al. 2010], and thus curb the use of these algorithms in design applications. Recently, virtual spherical lights [Hašan et al. 2009] were introduced to avoid the illumination loss from clamping, but at the expense of blurring some of the sharp reflections that provide important material cues. In this paper we introduce an alternative approach that alleviates these limitations and offers a better approximation of global illumination rendering with sharp glossy reflections.

Our approach is to split the light transport into two components: a dense global component and a sparse localized component. The global component of the light transport matrix corresponds to standard clamped instant radiosity and can be approximated well by techniques like matrix row-column sampling [Hašan et al. 2007]. We further observe that visibility can be separately (and often more coarsely) approximated compared to shading, particularly in scenes with glossy materials. We leverage this observation to construct a novel visibility clustering algorithm that requires fewer shadow maps than standard matrix row-column sampling. To approximate the local, high-rank component of the light transport, we introduce a scheme to trace *local* virtual lights from visible surfaces, and have these local lights correct the instant radiosity solution by compen-

sating for the lost energy through clamping. The time required to render the local lights is on the same order as for the global lights. This is in contrast with the path-traced compensation of Kollig and Keller [2004], where compensation dominates rendering in glossy scenes. The local lights illuminate small tiles of 32×32 pixels, and we are able to approximate their visibility, noting that most shadowing effects are already correctly handled in the global instant radiosity component of the solution. Thus by coupling visibility approximation for global lights and compensation through local lights we shade highly glossy materials efficiently.

Our global and local VPL technique with visibility approximations can achieve efficient rendering of scenes with glossy light transport in minutes for scenes with complex materials and lighting. The proposed solution could benefit applications where previewing of product appearance is critical and can have significant economic impact, for example in industrial design and architecture.

2 Related Work

Many previous global illumination methods have been designed without focus on glossy interreflection, and thus have problems when applied to glossy scenes. In many cases, the problem is that illumination is gathered from a discrete set of samples that has not been adapted to the requirements of the illumination receivers; in other techniques the adaptivity comes at a significant cost.

Non-adaptive methods. Rendering algorithms based on the instant radiosity formulation [Keller 1997] generate a number of virtual point lights (VPLs) to approximate indirect illumination, and shade visible surfaces from these lights. Matrix row-column sampling [Hašan et al. 2007] samples a small number of rows and columns from the large matrix of light-surface connections. Lightcuts [Walter et al. 2006] use a hierarchy on the set of lights and/or surface samples to improve the scalability of connection computations. Ritschel et al. [2008] use approximate shadow maps to accelerate rendering. Laine et al. [2007] propose incremental instant radiosity for animated sequences, where only a subset of shadow maps is recomputed in each frame.

In theory, many-light methods provide a correct, unbiased solution to the rendering equation; however, "spikes" (singularities) can be produced in parts of the image that strongly depend on a single VPL, since the BRDF and the geometry term can have a very large value compared to the light density. This is normally handled by *clamping*, which often removes interesting glossy illumination effects. The virtual spherical light (VSL) approach [Hašan et al. 2009] addresses the singularities and clamping requirement, but in some cases the density of the non-adaptively distributed VSLs is insufficient to reproduce sharp details of glossy reflection. This is also true for lightcuts variants, which do adaptively choose a subset of lights for shading, but the original discrete set itself is usually not sufficient for highly glossy effects.

The micro-rendering framework of Ritschel et al. [2009] provides interactive solutions and does not lead to singularities, but also has no mechanism to increase the density of samples in areas of spatial proximity or in glossy BRDF lobes, and cannot be easily extended to multiple bounces. Laurijssen et al. [2010] replaces sharp BRDFs on a cluster of camera path vertices by a smoother distribution, thereby reducing noise in indirect highlights.

Adaptive methods. Path tracing, bidirectional path tracing and Metropolis light transport [Veach 1997] use a number of strategies based on BRDF importance sampling to preferentially find paths with strong contributions to image pixels. These are unbiased approaches that can deliver the highest quality; however, they are wasteful in the sense that expensively constructed paths are usually not reused for many pixels. Kollig and Keller [2004] compensate for the clamped illumination in instant radiosity approaches by a recursive path tracing approach; however, in glossy scenes the compensation can be as slow as pure path tracing. Our local light approach is very similar in how it computes the compensation for clamping, but it is capable of doing so in time comparable to the clamped many-light rendering.

Segovia et al. [2006] introduced a many-light method that traces VPLs from the camera; this is related to our approach, but their VPLs are used globally and only applied to diffuse scenes. Path reuse [Bekaert et al. 2002] uses a similar concept to our local lights in a path-tracing context, but full visibility is checked for the connections used for path reuse, which limits possible speed-up. In contrast, our technique can afford to approximate the visibility for local lights, since it is computing only the compensation to a global pass that already handles most shadowing effects.

Photon mapping with final gathering [Jensen 2001] can utilize adaptive BRDF importance sampling from the camera, but the photon distribution itself is not adaptive, so large numbers of photons and nearest neighbors are required in glossy scenes. Progressive photon mapping [Hachisuka et al. 2008] addresses memory requirements of photon mapping by multi-pass processing, but does not fundamentally improve convergence.

Other related work. Our visibility clustering solution is similar to [Dong et al. 2009], which uses k-means on light positions and normals, while we use a data-driven approach based on a sparse sampling of visibility and shading. Arikan et al. [2005] decompose irradiance into near and far components, handling them by different approaches. This is related to our clamping-compensation decomposition, but does not consider glossy inter-reflection. Cheslack-Postava et al. [2008] introduce a precomputed method based on visibility approximation; their data-driven light tree construction is related to our visibility clustering algorithm.

3 Overview

In the path formulation of global illumination [Veach 1997], the illumination on a pixel j can be computed as an integral over all light paths in the scene passing through the pixel:

$$I_j = \int_\Omega f_j(\bar{x}) d\mu(\bar{x}),$$

where μ is a measure on the path space $\Omega = \bigcup_{k\geq 1} \Omega_k$, and Ω_k is the space of paths with k segments, $\bar{x} = x_0 x_1 \dots x_k$, such that x_0 is the camera position, x_1 is the surface point directly visible through the pixel, x_k is on a light source and $x_2, \dots, x_{k-1}$ are any light bounce points in the scene. The path contribution $f_j(\bar{x})$ is a product of BRDF and geometry terms on the vertices of the path, finally multiplied by light emission:

$$f_j(\bar{x}) = \left(\prod_{i=1}^{k-1} f_r(x_{i-1} \leftarrow x_i \leftarrow x_{i+1}) G(x_i \leftrightarrow x_{i+1}) \right) L_e(x_k \rightarrow x_{k-1}).$$

We will assume the scene is lit by a set of direct point light sources $\mathcal{L}_d$; area lights or environment maps can be handled by discretization to a large number of point lights. Under this assumption, paths of length 1 and 2 (emission and direct illumination) can be easily handled by summation over point lights, so the problem is reduced to computing the indirect component over paths of length 3 or more. Let $\Omega_{ind} = \bigcup_{k\geq 3} \Omega_k$. The indirect component I_j^{ind} can be computed by Monte Carlo integration, sampling N random paths and summing their contributions:

$$I_j^{ind} = \int_{\Omega_{ind}} f_j(\bar{x}) d\mu(\bar{x}) \approx \sum_{i=1}^{N} \frac{f_j(\bar{x}_i)}{\rho(\bar{x}_i)}. \quad (1)$$

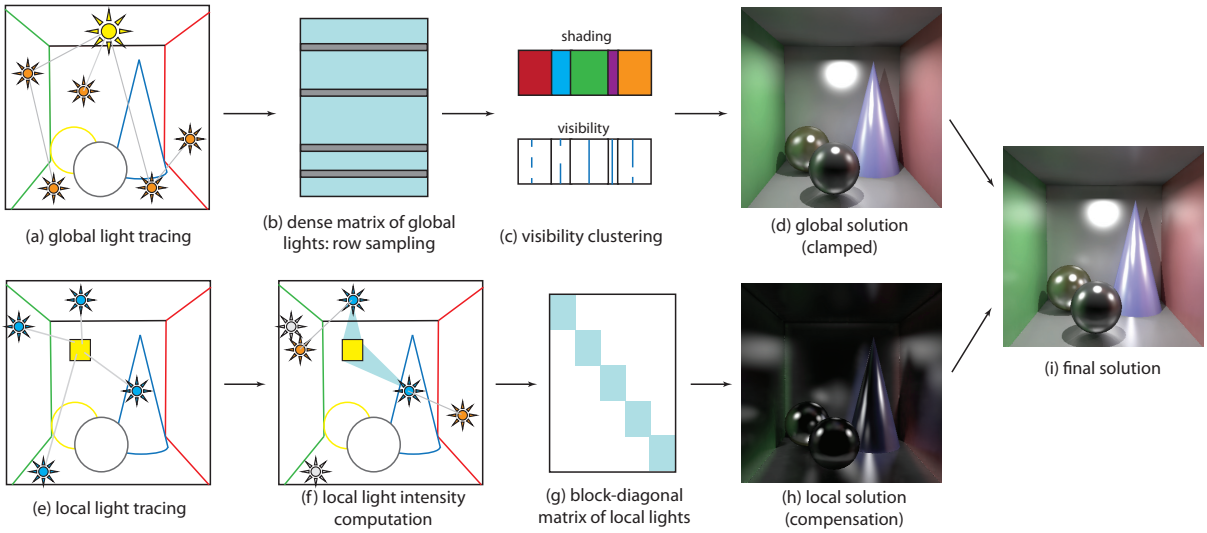

Figure 2: Conceptual overview of our algorithm. *Standard (global) virtual point lights are created by particle tracing (a), and a dense, clamped lighting matrix is assembled and row-sampled as in Hašan et al. [2007] (b). The reduced row matrix is separated into shading and visibity matrices; a clustering is found, with a binary visibility representative for every cluster (c). The shading is accumulated using the visibility representatives (d). Local lights are traced from image tiles (e) (here shown for one tile), and their intensities are computed by connection to global lights and probability density summation (f). This defines a sparse matrix of local light contributions (g) and is used to compute the clamping compensation (h). Adding (d) and (h) produces the final result (i).*

Here $\rho(\bar{x})$ is the path density (the number of paths per unit measure), and has to be positive for all paths with non-zero contributions. If all samples are independent and identically distributed with probability density $p(\bar{x})$, then we get the familiar $\rho(\bar{x}) = Np(\bar{x})$, though this does not have to be the case; for example, stratification can be applied.

Many-light methods. Algorithms based on the many-light approximation [Keller 1997] provide a sampling strategy for the above integral, by tracing sub-paths from direct light sources, treating these sub-paths as virtual point lights (VPLs), and connecting them to the visible surface samples (i.e. camera sub-paths of the form x_0x_1). This can conveniently be expressed as a lighting matrix $\mathbf{A}$ of light-sample contributions (i.e., the element $\mathbf{A}_{ij}$ will be the contribution of light j to sample i). In theory, this provides an unbiased solution to the global illumination problem; combined with the simplicity of the algorithm and the possibility of efficient implementations, this leads to the significant popularity of many-light methods in recent research.

Unfortunately, this sampling strategy is not always ideal, since the VPL density can be insufficient in corners and within glossy lobes. More precisely, the problem is that the path contribution $f_j(\bar{x})$ contains the following terms that have not been importance-sampled [Hašan et al. 2009]:

$$f_r(x_0 \leftarrow x_1 \leftarrow x_2)\ G(x_1 \leftrightarrow x_2)\ f_r(x_1 \leftarrow x_2 \leftarrow x_3) \tag{2}$$

If any of these terms is large (which often happens in corners and within glossy lobes), then a naïve application of the algorithm can cause disturbing artifacts. This is usually handled by clamping the terms in (2) to a user-specified constant c, and also by replacing the second BRDF term by a diffuse approximation (which we do not do in this paper). The clamping has an additional benefit of lowering the rank of $\mathbf{A}$, which improves the convergence of methods like row-column sampling and lightcuts, but the major drawback is that much glossy interreflection is lost. Virtual spherical lights use blurring rather than clamping to lower the rank of $\mathbf{A}$, which preserves illumination energy, but loses the clarity of glossy reflections.

Compensation. Compensating for the clamped illumination was proposed by Kollig and Keller [2004], but their solution is most efficient for diffuse scenes; in the presence of glossy BRDFs it is only marginally more efficient than pure path tracing. Instead, we notice that the low-rank, dense, *global* portion of the transport is handled well by the primary many-light technique, while the missing illumination tends to be sparse or *localized* in position-direction space. We propose to use a second many-light technique based on *local lights* to compensate. We define the clamping weights w_1 and w_2:

$$w_1(\bar{x}) = \min\left(1, \frac{c}{f_r(x_0 \leftarrow x_1 \leftarrow x_2)\ G(x_1 \leftrightarrow x_2)\ f_r(x_1 \leftarrow x_2 \leftarrow x_3)}\right)$$

and $w_2(\bar{x}) = 1 - w_1(\bar{x})$. The indirect illumination integral is thus split into the clamped part, I_j^1, and the compensation part, I_j^2:

$$I_j^1 = \int_{\Omega_{ind}} w_1(\bar{x}) f_j(\bar{x}) d\mu(\bar{x}) \quad I_j^2 = \int_{\Omega_{ind}} w_2(\bar{x}) f_j(\bar{x}) d\mu(\bar{x}) \tag{3}$$

We can compute I_j^1 by converting it to a dense, low-rank global light matrix $\mathbf{A}_g$ and applying any suitable many-light algorithm. In this paper, we use the method of *visibility clustering* (Section 4). To handle the compensation I_j^2, Section 5 introduces novel local lights to convert the problem into a sparse, high-rank local matrix $\mathbf{A}_l$. Figure 2 illustrates our system and Figure 3 gives a pseudo-code.

4 Visibility Clustering for Global Lights

In this section we introduce a data-driven visibility clustering algorithm that improves upon the matrix row-column sampling technique of Hašan et al. [2007], especially in cases where many column samples (and shadow map computations) are desired. Shadow map computations tend to be more expensive than shading, but shading is what needs to be sampled more densely for highly glossy materials. Therefore, we propose to partition the global lights into c clusters, render a single representative shadow map to approximate the *visibility* in each cluster, and combine that with the *shading* from *all* lights in the cluster.

Similar to [Hašan et al. 2007], we first sample a small subset of the rows of $\mathbf{A}_g$, resulting in the reduced matrix $\mathbf{R}$. We then optimize

the clustering on $\mathbf{R}$, and use the clustering for the full columns of $\mathbf{A}_g$. Due to the low-rank nature of $\mathbf{A}_g$, this produces good results.

Clustering objective. Each element of $\mathbf{R}$ expresses the contribution of a global light to a selected pixel, and each such contribution is the product of visibility and shading. Therefore, we can decouple the matrix into the visibility and shading components, $\mathbf{R} = \mathbf{V} \odot \mathbf{S}$, where $\odot$ denotes element-wise matrix multiplication. We will denote the columns of $\mathbf{V}$ and $\mathbf{S}$ as v_i and s_i. The key step in our algorithm is finding a clustering of the columns of $\mathbf{R}$ together with associated representatives from $\mathbf{V}$, such that the error of approximating the visibility within each cluster by the representative visibility is minimized.

Assume that a clustering $\mathcal{C} = C_1, \ldots, C_c$ is given, together with representative indices $r_1, \ldots, r_c$ in each cluster. We define the cost of a cluster as the error (in L_2-norm) incurred with the *optimal* representative. A light's *distance to the representative* can be defined as the error incurred by using the representative's visibility rather than the lights's own, i.e., $d_{i,r_p} = \|s_i \odot (v_i - v_{r_p})\|$; note that such a distance measure is non-symmetric. To find the optimal clustering, we want to minimize the sum of squared distances from each light to its representative. More formally, the cost of a clustering can be expressed as:

$$cost(\mathcal{C}) = \sum_{p=1}^{c} cost(C_p) = \sum_{p=1}^{c} \sum_{i \in C_p} \|s_i \odot (v_i - v_{r_p})\|^2. \quad (4)$$

This clustering problem is related to the k-means and k-median problems, but with a different, non-symmetric distance measure.

Clustering algorithm. We use a hierarchical partitioning approach: starting with all lights in a single cluster, we keep splitting the cluster with currently largest cost, until the desired number of clusters is reached. To split a cluster into two, we choose the column that is farthest from the current representative as a second representative, and iterate the following two steps, each of which is guaranteed not to increase the objective function:

1. Create two partitions by assigning each column to the closer of the two representatives based on the non-symmetric distance d_{i,r_p}

2. Pick an optimal representative for the two partitions. (A representative of a cluster is optimal if it minimizes the cluster cost.)

We could iterate until convergence, but we found that 1-2 iterations are sufficient to find a good cluster split. To pick the optimal representative efficiently, we leverage the fact that visibility is a binary function. For each row of the cluster submatrix, we compute two numbers: the errors incurred by approximating the visibility in the row by 0 and by 1; this can be done in a linear pass. In a second linear pass, we evaluate the error of any representative by picking and summing the appropriate error values for each row.

Due to the representative refinement step, the hierarchical partitioning strategy is slowest in the beginning, when spliting very large clusters. We improve performance by using the position of the global lights for the first 16 splits, roughly doubling clustering speed with no negative impact. We also parallelize the hierarchical splitting algorithm, by keeping a thread-safe priority queue of the current clusters, and using multiple worker threads for splitting.

Final rendering. Once the clustering have been found, we render the global lights by iterating over the visibility clusters. For each cluster, we compute the shadow map for the cluster's representative, query it to produce a *shadow mask* that specifies a visibility value for each pixel, and accumulate the shading for each visible pixel from all lights in the cluster (unlike row-column sampling, which uses both visibility and shading only from the representative).

5 Local Lights

Recall that we want to find a technique to compute the component I_j^2 (defined in Equation 3) missing (clamped away) from the global solution. Intuitively, as the global solution leads to a dense, low-rank matrix, one would expect that the compensation problem can be formulated as a high-rank, sparse matrix, and that an algorithm should exist that takes advantage of this particular structure. Note that this algorithm has to be a complete global illumination solution (we could set $w_1 = 0$ and $w_2 = 1$, leaving all work to the compensation), but ideally we would like it to be well adapted to handling exactly the localized illumination effects that remain in I_j^2. We have found such a technique, based on the concept of *local lights*.

Derivation of local lights. Consider a simple gathering algorithm that would compute the compensation I_j^2 by tracing r rays using BRDF importance sampling at point x_1, connecting each hitpoint x_2 to a single global light g by importance sampling according to power, and scaling the contribution by the clamping compensation weight $w_2(\bar{x})$. The global lights already handle multiple indirect bounces; therefore, so does this gathering algorithm. This is a variation of the Kollig-Keller approach [2004].

Note that the ray hitpoints x_2 could be thought of as "local lights" that contribute their illumination to only a single pixel: the one they were sampled from. This thought experiment suggests the wastefulness of the technique; would it not be better to contribute the illumination to neighboring pixels as well, thus amortizing the effort in creating the local light?

We consider a small image tile of size $t \times t$, where $t = 32$ is a typical value, and we let all the local lights that originate from this tile contribute to all pixels in the tile. The key challenge is to define the contribution of such a local light at position x_2 to any tile pixel x_1. Using equation (1), all we need is to define the contribution and density of the imaginary path $\bar{x} = x_0 x_1 x_2 x_g$, where x_g is the position of the global light g. We find that this contribution will be:

$$\frac{w_2(\bar{x}) f_r(x_0 \leftarrow x_1 \leftarrow x_2) G(x_1 \leftrightarrow x_2) f_r(x_1 \leftarrow x_2 \leftarrow x_g) E_g(x_2)}{\rho(x_2) p_g},$$

where $E_g(x_2)$ is the irradiance due to the global light g at point x_2, p_g is the discrete probability of choosing g out of all global lights, and $\rho(x_2)$ is the density of local lights generated from the current tile, computed as the aggregate density of generating a local light at x_2 by BRDF sampling from all pixels in the tile:

$$\rho(x_2) = \sum_{x_1 \in \text{tile}} \rho(x_0 \rightarrow x_1 \rightarrow x_2).$$

The density $\rho(x_0 \rightarrow x_1 \rightarrow x_2)$ depends on the exact distribution used for BRDF importance sampling (which may or may not precisely match the BRDF) and on the number of samples taken.

Visibility. We ignore visibility computation in the above equations, both in the geometry term $G(x_1 \leftrightarrow x_2)$ and in the density term $\rho(x_0 \rightarrow x_1 \rightarrow x_2)$. Note that x_2 is necessarily visible from the point x_1 where the local light originated, and other points x_1' in the tile will usually also be visible from x_2, especially over short distances (and over long distances we often have $w_2 = 0$ anyway). This assumption causes very few problems and allows for an efficient computation of local lights on the GPU. Note that a local light can sometimes contribute to a part of the tile that is quite distant from the pixel where it originated, though in practice the compensation weight w_2 will often be very low or zero in such situations, and we have not observed problems caused by this.

Rejection. Local lights often end up in areas that have not been clamped (where $w_1 = 1$), and will have $w_2 = 0$ and therefore zero contribution. However, checking if w_2 is indeed zero over

```
def render(scene, opts):
    img = zero_img(opts.img_size)
    dfb = create_deep_framebuffer(scene, opts.img_size)
    global_lights = scene.direct_lights() + trace_indirect_lights(scene)
    render_global_component(scene, opts, dfb, global_lights, img)
    render_local_component(scene, opts, dfb, global_lights, img)
    return img

def render_global_component(scene, opts, dfb, global_lights, img):
    # create reduced shading and visibility matrices by row sampling
    S = zeros(opts.num_rows, len(global_lights))
    V = zeros(opts.num_rows, len(global_lights))
    row_pixels = choose_random_pixels(dfb, opts.num_rows)

    for pixel in row_pixels:
        shading, visibility = render_row_on_gpu(pixel, global_lights)
        S(pixel.index, :) = shading
        V(pixel.index, :) = visibility

    # visibility clustering
    clusters = [initial_cluster(global_lights, S, V)]

    while len(clusters) < opts.num_clusters:
        c = extract_highest_cost_cluster(clusters)
        c1, c2 = split(c) # use splitting algorithm from section 4
        clusters += [c1, c2]

    # render clusters
    for c in clusters:
        vis_mask = render_visibility_on_gpu(dfb, c.rep)
        for light in c.items:
            shading = render_shading_on_gpu(dfb, light, opts.clamp)
            img += vis_mask * shading

def render_local_component(scene, opts, dfb, global_lights, img):
    for pixel in dfb.pixels:
        for i in range(0, num_local_lights_per_pixel):
            # create local light by BRDF importance sampling
            direction = pixel.sample_brdf()
            light = trace(pixel.position, direction)
            if light == None: continue

            # connect to global light, to determine the intensity
            # of the local light and its incoming direction
            intensity, incoming = connect(light, global_lights)

            # local light rejection
            clamped_term = fr(camera <- pixel <- light) * G(pixel, light) *
                fr(pixel <- light <- incoming)
            if clamped_term < opts.clamp / 2 : continue

            # choose a tile that contains this pixel
            tile = choose_random_offset_tile(pixel)

            # find the density (in area measure) of this light, accounts for rejection
            rho = 0
            for p in tile.pixels: rho += eval_area_pdf_on_gpu(p, light)

            # compute contribution to tile pixels
            light.incoming = incoming
            light.intensity = intensity / rho
            for p in tile.pixels:  img[p] += shade_w2_on_gpu(p, light, opts.clamp)

def connect(local_light, global_lights):
    # for simplicity, we show a single power-sampled connection; other schemes are
    # possible (we use Kollig-Keller compensation in Tableau, Disney, and Kitchen 2)
    global_light, prob = sample_power(global_lights)
    intensity = shade(local_light, global_light) / prob
    return intensity, normalize(global_light.pos - local_light.pos)
```

Figure 3: Pseudo-code of our algorithm.

the whole tile would be expensive; we instead use the following heuristic: if w_2 would have been zero for the generating pixel even if the clamping constant c was halved, the light is rejected. Note that this does not introduce bias, but care must be taken to correctly define $\rho(x_0 \to x_1 \to x_2)$; it is the density of lights created at x_2 by sampling from x_1 that would *not* have been rejected.

Reusing samples traced from the camera has been explored in [Segovia et al. 2006] and [Bekaert et al. 2002]; our local lights are distinguished by several features, including their use only for the compensation I_j^2, the possibility of rejection in areas where compensation is not necessary, and their locality and visibility approximation that allow for efficient GPU implementation.

6 Implementation Details

This section lists some implementation improvements to the performance and image quality of the algorithm.

Local light clamping. In scenes with glossy surfaces very close to each other, the occasional "spikes" (singularities) can still appear even with local lights. To prevent the resulting artifacts, a small amount of clamping can be applied to the contribution of a local light to pixels and/or the connection to the global light. Connection clamping can be compensated by the Kollig-Keller technique [2004], applied to the local-global connection. We use this technique in the Tableau, Disney and Kitchen 2 scenes, using 3 – 10 compensation rays per local light. This removes the bias introduced by connection clamping at the expense of additional ray tracing.

Jittered tiling. Subdividing the image into fixed tiles has the disadvantage that tile boundaries can be perceptible. We overcome this problem by randomly choosing, for each local light, a corresponding tile that contains the pixel where the local light originated. Furthermore, we can also weight the contributions to tile pixels by any kernel that preserves energy; we use a linear ramp on the tile edges to render them even less perceptible. These techniques improve the rendering quality while not introducing bias into the result; the only bias in the local light method comes from ignoring visibility and their optional clamping.

Antialiasing. For the local lights, as well as for larger visibility clusters, we employ interleaved sampling antialiasing, where each light contributes to only a single subpixel within a pixel.

7 Results

In this section we first provide insight into our methods for each transport matrix component individually. Next, we show how the strengths of the two methods complement each other, resulting in fast generation of high quality images. We then show results for four scenes, each with its glossy-glossy interaction challenges, and compare them with path tracing and the virtual spherical light (VSL) technique from [Hašan et al. 2009]. The image sizes are 800×600 and use 3×3 anti-aliasing with interleaved sampling. All our measurements are done on a system with two Intel Xeon X5560 processors (with 4 cores each), 8 GB RAM, and an NVIDIA GTX 480 graphics card. Path traced images were generated on a cluster of 16 nodes, where each node was a dual Xeon 2.83 GHz (4-core) machine. Reported times for path traced images are the time using only one node of the cluster (computed by adding the times on all cluster nodes together).

Figure 5 shows a detail of our **Tableau** scene (for the full image see Figure 8) rendered with matrix row column sampling (MRCS) and our method, rendered in approximately the same time. It clearly demonstrates the extra image quality our method is able to achieve in scenes with curved glossy surfaces because we are able to use more shading samples, even though we use fewer shadow maps for visibility. However, we note that visibility clustering and MRCS

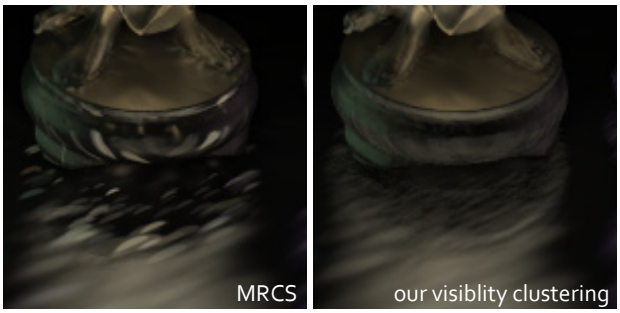

Figure 5: Global light methods. *Comparison of matrix-row column sampling (left: 202 seconds) and visibility clustering (right: 213 seconds). Both methods use 200,000 VPLs, and compute only indirect illumination. MRCS selects 10k lights for shadow map evaluation and shading, while visibility clustering uses only 5k shadow maps and shades from all VPLs, yielding less splotchy images.*

Figure 4: Component separation. *Left: the long distance effects captured by global lights; notice the shadows behind bottles on the counter. Center: we show the local glossy-glossy interaction, mainly between the back wall and the towel rack and faucet just in front of it. Right: combination of both components, resulting in an image with both local and global effects. (The images show indirect illumination only.)*

Figure 6: Local lights only. *We observe that when only local lights are used, fine shadows may disappear (e.g., there are no shadows behind the bottles).*

can be used together, if the difference between the ranks of the visibility and shading matrices is low. In such a setup the visibility clusters are used as an initial clustering for MRCS, which can then further refine them into the desired number of shading clusters. However, we do not use this approach, because it brings almost negligible speedup when the method is paired with local light rendering, since local light generation and rendering accounts for a large (approximately 50%) fraction of total rendering time.

The sparse high-rank component of the transport matrix is handled by the newly proposed local light algorithm. While this algorithm excels in capturing local glossy interactions, Figure 6 shows the limitations of the method as a stand-alone solution. Due to the local visibility approximation, all shadows smaller than the tile size are effectively smeared away (compare with Figure 4).

While global methods lack local glossy effects and local lights lack fine shadows, combining both we achieve the desired high quality images. Figure 4 shows a typical glossy scene (**Kitchen 1**) separated into its global and local components. Global lights provide the basic illumination and shadows, but the far wall appears to be diffuse. Local lights correct this incorrect perception of material properties by providing glossy reflections of the paper towels and faucet, and of course all the other missing local interactions. These two components are combined for the final result image. Note that all three images show indirect illumination only.

Figure 7 shows the **Kitchen 1** scene (253,433 triangles) rendered using our method, path tracing, and VSLs for comparison. We also show insets of areas where our technique is able to accurately compute reflections that contribute to material perception (where VSLs miss those features). In fact, VSLs give results similar to the solution using only global lights in Figure 4. The VSL approach misses reflections of the towel rack, faucet, and the yellow plates on the back wall, the bar stool rods on the base of the stools, and other reflections on the counter. The two color-coded error images show that the overall light distribution of our method matches the reference more closely. However, we can notice that the illumination at the right side of the glossy wall is not perfectly even. This is result of slight noise in connection strategy for local lights.

Figure 8 shows the **Tableau**, **Disney**, **Kitchen 2** scenes from [Hašan et al. 2009]. Note that we have increased the gloss of the floor in **Tableau** (compared to the original paper) to demonstrate our ability to render high gloss accurately. We also provide comparison with Progressive Photon Mapping [Hachisuka et al. 2008] and Stochastic Progressive Photon Mapping [Hachisuka and Jensen 2009] as suplementary material.

In **Tableau**, our technique is able to capture both the long-range and local reflections when compared to VSLs. In fact, the VSL reflections are quite blurry, and can cause incorrect material perception [Ramanarayanan et al. 2007; Křivánek et al. 2010]. In comparison with the path tracer, our solution in this difficult scene is quite accurate. In **Disney** our algorithm is able to capture the two blue caustics (see insets) and the shape of highlights accurately when compared with the VSL approach. In **Kitchen 2** our technique accurately computes reflections of the cupboards in the back. However, while it captures reflection of the pot in the front, this reflection is darker. Similar darkening is also perceivable in **Disney** on the left wall. Both are caused by application of slight clamping on local light contribution.

Table 1 gives a detailed breakdown of individual stages of our algorithm. To fully utilize the computing power, we run the local light generation on the CPU concurrently with the global and local light rendering on the GPU. The local rendering time therefore also includes the time required to finish local light generation.

	Kitchen 1	Tableau	Disney	Kitchen 2
# global lights	300k	200k	200k	100k
# vis. clusters	10k	5k	15k	10k
# local lights	17.1M	55.6M	13.5M	25.1M
Row render	18.6 s	22.6 s	11.3 s	15.4 s
Vis. clustering	19.0 s	13.0 s	28.6 s	12.5 s
Global render	249.6 s	145.6 s	90.8 s	170.5 s
Local render	40.6 s	162.1 s	33.0 s	57.5 s
Total	**327.8 s**	**343.3 s**	**163.7 s**	**255.9 s**

Table 1: *Number of lights and visibility clusters (top) and breakdown of the time spent on different parts of the algorithm (bottom).*

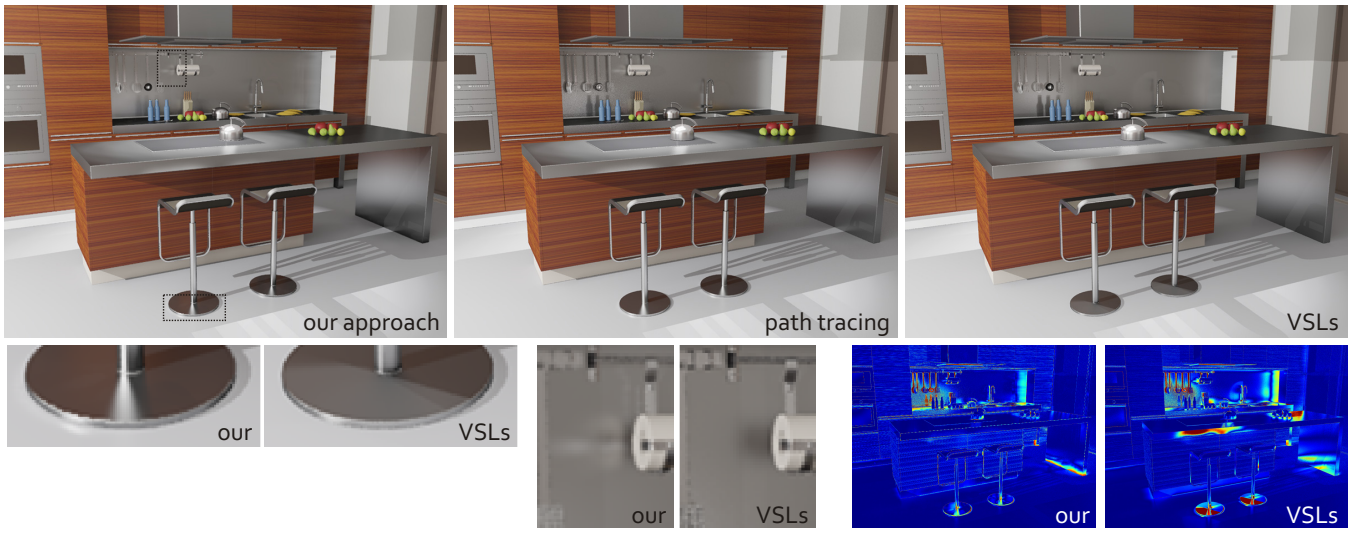

Figure 7: Kitchen 1. *Top-right: Our approach (5 min 28 sec). Top-middle: Reference path traced solution (still noisy, 106 hours). Top-right: Virtual Spherical Lights (VSLs), after 6 min 25 sec. Insets in the bottom row show some of the glossy interactions captured by our method that are missing from the VSL rendering. Bottom-right: Color-coded relative error images of our method and VSL against the reference solution.*

Antialiasing. Our interleaved antialiasing scheme enables the efficient computation of antialiased images with less than a 2× cost in performance. For example, for **Kitchen 1**, the regular image takes approximately 130 seconds, while the 3×3 anti-aliased image takes approximately 330 seconds to compute. All results presented are for antialiased images.

8 Conclusion

We present a component-based rendering algorithm for rendering of highly glossy materials with global illumination. Our approach is to split the light transport into two parts: a global, low-rank component, and a sparse, localized, high-rank component. We approximate the low-rank component using a novel visibility clustering algorithm, and the high-rank component by using local virtual lights to compensate for lost energy due to clamping. Our solution is suitable for previewing in industrial design applications where materials like metals and plastics are common and should be reproduced with high fidelity.

Limitations and Future Work. While this approach expands the range of materials that can be rendered accurately, it has a few limitations. Local VPLs still need clamping when there are highly glossy or close-range interactions. Fine shadows can be dulled due to the visibility approximation in the local lights component. The number of local VPLs required is highly scene dependent; a progressive algorithm could be designed for the local component. Future work could explore automated selection of parameters and clamping. Clustering and virtual light tracing is currently done on the CPU; a full GPU solution could increase performance.

Acknowledgements

This work was supported by a Marie Curie Fellowship PIOF-GA-2008-221716 within the 7th European Community Framework Programme, NSF CAREER 0644175, NSF CPA 0811680, and grants from Intel Corporation and Microsoft Corporation.

References

ARIKAN, O., FORSYTH, D. A., AND O'BRIEN, J. F. 2005. Fast and detailed approximate global illumination by irradiance decomposition. *ACM Trans. Graph. 24*, 3, 1108–1114.

BEKAERT, P., SBERT, M., AND HALTON, J. 2002. Accelerating path tracing by reusing paths. In *Eurographics Workshop on Rendering*, 125–134.

CHESLACK-POSTAVA, E., WANG, R., AKERLUND, O., AND PELLACINI, F. 2008. Fast, realistic lighting and material design using nonlinear cut approximation. *ACM Trans. Graph. 27*, 5, 128:1–128:10.

DONG, Z., GROSCH, T., RITSCHEL, T., KAUTZ, J., AND SEIDEL, H.-P. 2009. Real-time indirect illumination with clustered visibility. In *Vision, Modeling, and Visualization Workshop 2009*.

HACHISUKA, T., AND JENSEN, H. W. 2009. Stochastic progressive photon mapping. *ACM Trans. Graph. 28*, 5, 1–8.

HACHISUKA, T., OGAKI, S., AND JENSEN, H. W. 2008. Progressive photon mapping. *ACM Trans. Graph. 27*, 5, 130:1–130:8.

HAŠAN, M., PELLACINI, F., AND BALA, K. 2007. Matrix row-column sampling for the many-light problem. *ACM Trans. Graph. 26*, 3, 26:1–26:10.

HAŠAN, M., KŘIVÁNEK, J., WALTER, B., AND BALA, K. 2009. Virtual spherical lights for many-light rendering of glossy scenes. *ACM Trans. Graph. 28*, 5, 143:1–143:6.

JENSEN, H. W. 2001. *Realistic image synthesis using photon mapping*. A. K. Peters, Ltd., Natick, MA, USA.

KELLER, A. 1997. Instant radiosity. In *Proc. SIGGRAPH 97*, 49–56.

KOLLIG, T., AND KELLER, A. 2004. Illumination in the presence of weak singularities. In *Monte Carlo and Quasi-Monte Carlo Methods*, 245–257.

KŘIVÁNEK, J., FERWERDA, J. A., AND BALA, K. 2010. Effects of global illumination approximations on material appearance. *ACM Trans. Graph. 29*, 4, 112:1–112:10.

LAINE, S., SARANSAARI, H., KONTKANEN, J., LEHTINEN, J., AND AILA, T. 2007. Incremental instant radiosity for real-time indirect illumination. In *Eurographics Symposium on Rendering*, 277–286.

LAURIJSSEN, J., WANG, R., DUTRÉ, P., AND BROWN, B. J. 2010. Fast estimation and rendering of indirect highlights. *Computer Graphics Forum 29*, 4, 1305–1313.

RAMANARAYANAN, G., FERWERDA, J., WALTER, B., AND

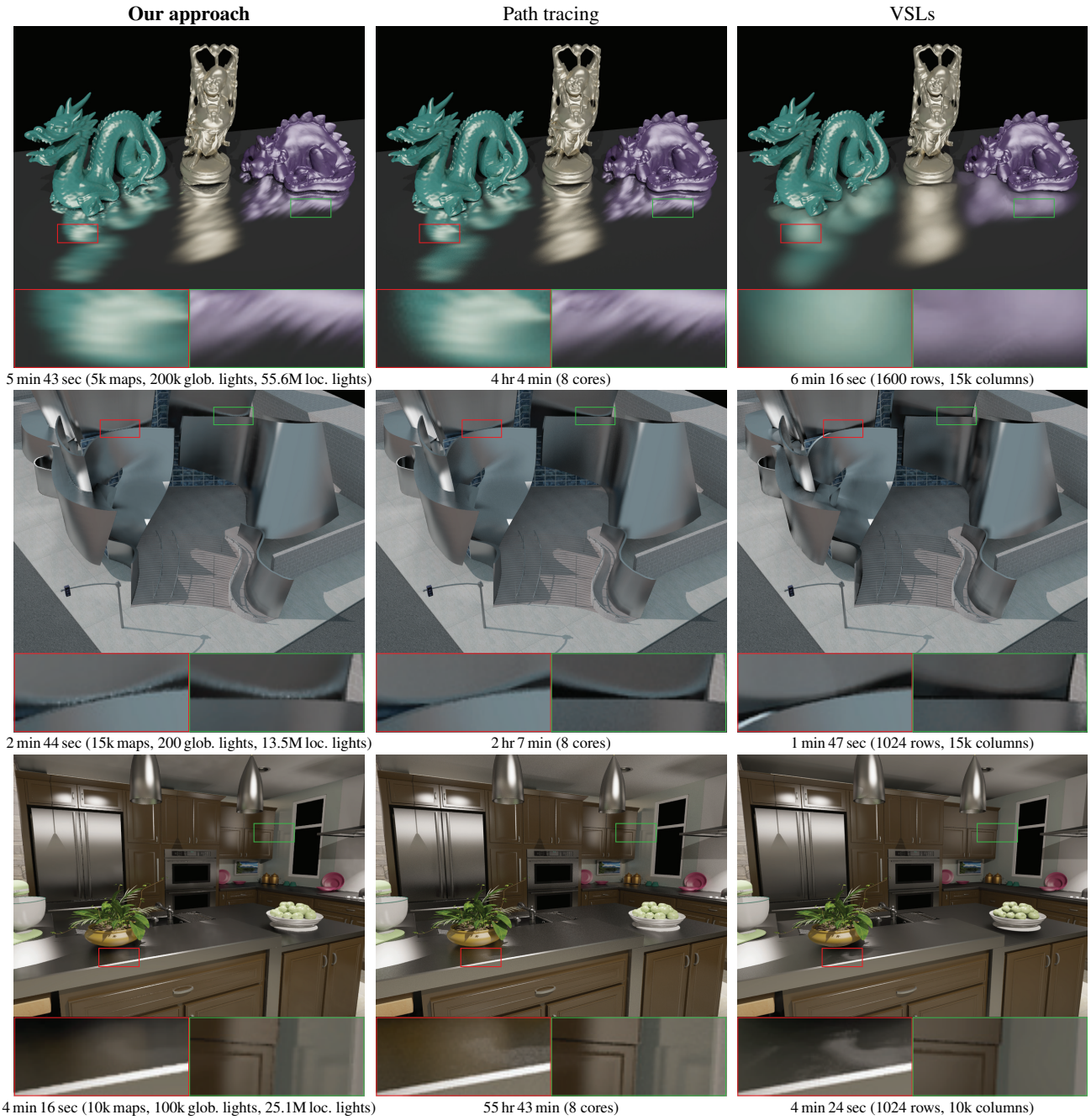

Figure 8: Results: Tableau, Disney, Kitchen 2 *from Hašan et al. [2009] and comparisons with path tracing and VSLs. Note that the Tableau scene has higher gloss than in the original paper. Corresponding times, shadow map numbers, and number of global and local lights, are also reported. Insets show where our technique captures lighting features that match the path tracer, but are missing in the VSL solutions.*

BALA, K. 2007. Visual equivalence: towards a new standard for image fidelity. *ACM Trans. Graph. 26*, 3, 76:1–76:11.

RITSCHEL, T., GROSCH, T., KIM, M. H., SEIDEL, H.-P., DACHSBACHER, C., AND KAUTZ, J. 2008. Imperfect shadow maps for efficient computation of indirect illumination. *ACM Trans. Graph. 27*, 5, 129:1–129:8.

RITSCHEL, T., ENGELHARDT, T., GROSCH, T., SEIDEL, H.-P., KAUTZ, J., AND DACHSBACHER, C. 2009. Micro-rendering for scalable, parallel final gathering. *ACM Trans. Graph. 28*, 5, 132:1–132:8.

SEGOVIA, B., IEHL, J.-C., AND PÉROCHE, B. 2006. Bidirectional instant radiosity. In *Eurographics Symposium on Rendering*, 389–398.

VEACH, E. 1997. *Robust Monte Carlo Methods for Light Transport Simulation*. PhD thesis, Stanford University.

WALTER, B., ARBREE, A., BALA, K., AND GREENBERG, D. P. 2006. Multidimensional lightcuts. *ACM Trans. Graph. 25*, 3, 1081–1088.

WANG, R., WANG, R., ZHOU, K., PAN, M., AND BAO, H. 2009. An efficient GPU-based approach for interactive global illumination. *ACM Trans. Graph. 28*, 3, 91:1–91:8.