Fast Light-Map Computation with Virtual Polygon Lights

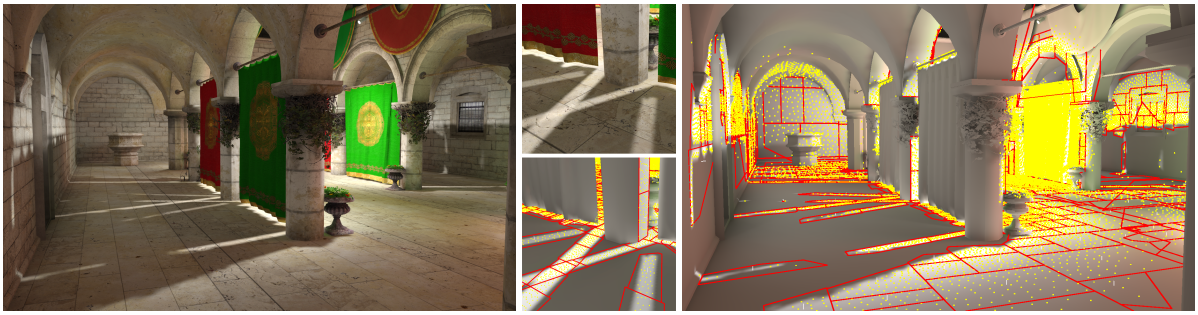
Christian Luksch* VRVis Research Center Austria

Robert F. Tobler† VRVis Research Center Austria

Ralf Habel‡ Vienna UT / Disney Research Zürich

Michael Schwärzler§ VRVis Research Center Austria

Michael Wimmer¶ Vienna University of Technology



Scene illuminated from 9 light sources rendered with a 16MP light map computed from 5,300 virtual lights in 30s.

Abstract

We propose a new method for the fast computation of light maps using a many-light global-illumination solution. A complete scene can be light mapped on the order of seconds to minutes, allowing fast and consistent previews for editing or even generation at loading time. In our method, virtual point lights are clustered into a set of virtual polygon lights, which represent a compact description of the illumination in the scene. The actual light-map generation is performed directly on the GPU. Our approach degrades gracefully, avoiding objectionable artifacts even for very short computation times.

CR Categories: Computer Graphics [I.3.7]: Three-Dimensional Graphics and Realism—Color, shading, shadowing, and texture

Keywords: global illumination, light-maps, instant radiosity

1 Introduction

Global illumination methods can be categorized into two classes: high-quality offline solutions, which often take hours to compute; and interactive solutions, which take some shortcuts to increase performance. However, interactive solutions typically require significant hardware resources, restricting their use to high-end GPUs, and often exhibit noise and flickering artifacts, especially when the camera is moving.

When restricting the computation to view-independent illumination effects, it is possible to precompute the lighting in the scene into so-called *light maps*. These are still very popular despite the restriction to view-independent lighting, because they allow the scene to be rendered at full frame rates at little hardware resources, and completely avoid noise and flickering artifacts.

Unfortunately, the computation of light maps is again an offline process, preventing dynamic changes in lighting or the scene. In applications such as level design for games, architectural lighting design, interior design and many others, it is important to be able to change the lighting situation while still being able to provide a real-time and artifact-free walkthrough as afforded by light maps.

In this paper, we therefore present a method to accelerate the computation of light maps to the level of seconds instead of hours, so changes in the lighting situation can still be observed in an interactive application after a short delay, without requiring a full offline computation cycle. This makes the method applicable for lighting design applications, game level design, or even in-game computation, where lighting for a level can be defined and computed at loading time, therefore avoiding having to store the full lighting solution on the distribution medium.

In order to achieve near-interactive precomputation speeds for light maps, we build on the concept of virtual point lights, which have been used in interactive global illumination before. As an improvement we introduce the idea of *virtual polygon lights*, which allow a much higher quality solution than previous interactive methods. In particular, we offer the following contributions:

- A novel VPL clustering method for generating virtual polygon lights providing a good fit especially for planar geometry.
- An evaluation of the virtual polygon lights that uses analytical form factors, providing higher accuracy and less artifacts.
- Independent of the concrete illumination algorithm used, we also show a new rasterization method for light maps that ensures that no light-map texels are omitted.

2 Related Work

Only few publications focus on light-map creation, which poses a different problem than images synthesis. Though the same principal approaches can be taken in both cases, there are large differences in the applicability of a method to the two differing problems due to calculation time and memory requirements.

*e-mail:luksch@vrvis.at
†e-mail:tobler@vrvis.at
‡e-mail:ralf.habel@disneyresearch.com
§e-mail:schwaerzler@vrvis.at
¶e-mail:wimmer@cg.tuwien.ac.at

Initially, light maps were often generated using radiosity [Cohen et al. 1993], since this solves the global illumination for the complete scene independent from the viewpoint, but in principle, any general global-illumination method such as path-tracing [Kajiya 1986] or bi-directional path-tracing [Lafortune and Willems 1993] can be used for light mapping as well. However, the complete lighting in large scenes usually requires very long rendering times using these general approaches.

Photon mapping [Jensen 1996], with optimizations such as clustered photon gathering [Wang et al. 2009], parallel final gathering through micro-rendering [Ritschel et al. 2009], image-space photon mapping [McGuire and Luebke 2009], interactive global photon mapping [Fabianowski and Dingliana 2009] and parallel progressive photon mapping [Hachisuka and Jensen 2010] can create single realistic images in the order of seconds. Due to the view-independence, photon mapping can be used directly to calculate light maps, though storing and gathering a large amounts of photons may be prohibitively expensive in large scenes.

Recently, parallel global ray-bundles [Hermes et al. 2010] have been used to generate light maps in a couple of minutes [Tokuyoshi et al. 2011]. Voxel-based global illumination techniques (e.g. [Thiedemann et al. 2011] or [Crassin et al. 2011]) can perform an approximate light transport even for dynamic objects interactively. Since voxels are normally not aligned to the geometry, they may lead to objectionable artifacts such as light leaks or re-rasterization problems and are therefore very problematic for high-quality light-map generation.

Many-light global illumination approaches have their origin in instant radiosity introduced by Keller [1997] and efficiently capture the global light distribution even in large scenes using virtual lights. For better scalability, Walter et al. [2005] introduced lightcuts that generate a clustered hierarchy of virtual lights and choose an appropriate cut through the light-tree per pixel, based on an error metric to significantly reduce rendering costs. Furthermore, matrix row-column sampling [Hašan et al. 2007] allows to approximate visibility calculations for virtual lights by GPU rasterization.

Most methods use virtual point lights (VPLs) which lead to illumination spikes and energy loss due to clamping, introducing major artifacts and loss of physicality in many situations. Recently, Hasan et al. [2009] introduced spherical virtual lights to avoid the point light singularities. Novak et al. [2011] approach this problem by compensating the energy loss in screen-space as a post-processing effect.

In real-time implementations, VPLs are often generated in a reflective shadow mapping (RSM) [Dachsbacher and Stamminger 2005] pass from a rasterized light-view, capable of generating thousands of VPLs instantaneously. Shadow mapping [Williams 1978] is used for visibility computations. Since shadow mapping is the limiting factor, Laine et al. [2007] reduce the number of shadow map updates with a caching and update strategy. Ritschel et al. [2008] speed up the visibility computations using imperfect shadow maps (ISM), allowing visibility tests from hundreds of points and thereby enabling interactive computation of multiple indirect bounces. View-dependent optimizations of ISM even allow large dynamic scenes [Ritschel et al. 2011]. Dong et al. [2009] perform a VPL-clustering to reduce the number of visibility tests. To account for visibility artifacts, an area approximation of the clustering is used to generate soft shadows with convolution soft shadow maps (CSSM) [Annen et al. 2008]. A different clustering and area light evaluation has been used by Prutkin et al. [2012] for a real-time single bounce global illumination.

3 Overview

To produce high-quality light maps, we need to calculate the complete lighting of the whole scene in the texture space of the scene geometry. Figure 1 shows the components and data flow of our approach. As input of the light map computation, we assume a static scene with a complete, unambiguous uv-parametrization suitable for a light map atlas (see Figure 2).

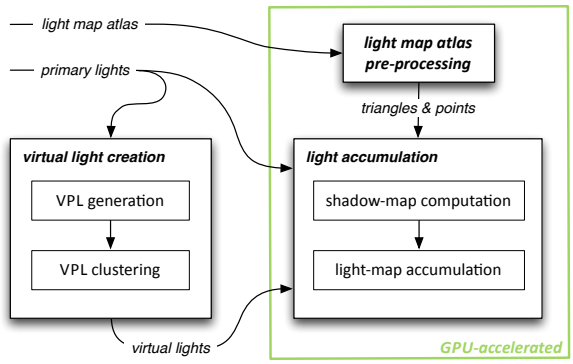


Figure 1: *Overview of the proposed method.*

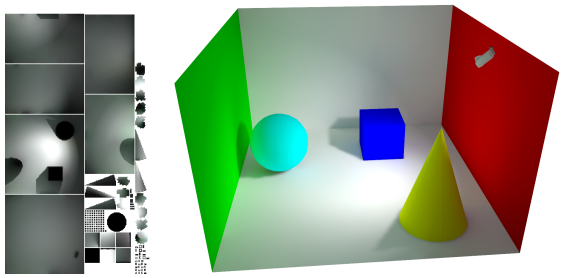


Figure 2: *Example of a light map atlas (left) with baked global illumination and the light mapped scene (right).*

In a pre-processing step, the *light-map atlas pre-processing*, described in Section 4, we use the GPU to quickly identify light map texels that would be missed if only standard triangle rasterization is used for light map calculation. This step has to be carried out only once, as long as the light map parameterization does not change, i.e. when objects are only rigidly transformed. On the CPU(s) we perform *virtual light creation* (see Section 5), starting with a large number of VPLs generated by tracing rays from the primary light sources. These VPLs are clustered first into planes, and within each plane into similar sized clusters using a kd-tree in order to create a much smaller number of higher-order virtual lights. On the GPU we perform *light accumulation* (see Section 6) by calculating illumination for each light-map texel with shadow mapping from the primary lights and all generated virtual lights.

4 Light-Map Atlas Pre-processing

In order to accumulate the illumination of virtual lights, we need to (1) identify all light-map texels that might later be accessed by the renderer, and (2) provide a method to enumerate all those texels together with their world-space positions and normals to calculate the illumination for each texel. By rasterizing the scene geometry directly into the light-map atlas using the texture coordinates as positions, the mapped light map texels can be identified, and corresponding world-space positions and normal vectors can be calculated by interpolation.

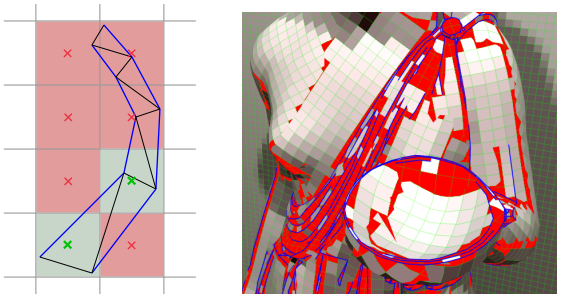


Figure 3: *Continuously parametrized geometry rasterized in uv-space on the GPU (left). Due to rasterization rules only texels whose center falls within a triangle are generated (green). Light mapped scene geometry (right) accessing empty texels (red). Border edges where parametrization is forced to be separated (blue).*

Interestingly enough, this holds a surprising challenge for light map rendering if these steps are performed on the GPU using the standard rasterization pipeline: due to the rasterization rules, only texels whose centers fall within the analytical texture-space triangle actually produce a fragment, as illustrated in Figure 3 (left). Since any texel intersected by the triangle edges can be queried when the light-mapped scene is rendered, illumination information might be missing (red texels in Figure 3, right). In the worst case, a parameterized geometry part may hit few or even no texel centers of the assigned region, making dilation strategies too inaccurate and therefore unfeasible. This problem could be countered by

- analyzing the parameterization and adjusting the uv-coordinates (complex cases, error prone),
- accumulation of multiple varying rasterizations (high overhead) or
- conservative rasterization [Hasselgren et al. 2005] (expensive),

or completely avoided by pre-computing world-space positions and normals of all texels into additional buffers (high memory requirements).

In order to avoid the disadvantages of these methods, we propose a fast solution with low memory consumption that avoids repeated drawing of texels: since point primitives are guaranteed to be rasterized, we use an additional list of points to generate the world-space positions and normals of texels missed by the triangle rasterization. The detection of missed texels and the consecutive generation of this point list only needs to be performed once in a pre-compution step, and also detects triangles that do not by themselves (i.e., without the rendering of edge or point primitives) produce any texels and can therefore be omitted in future rasterizations.

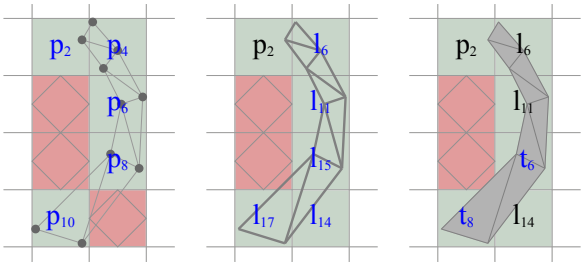


Figure 4: *Combining rasterization rules of point (left), lines (middle) and triangles (right) in three passes allows reconstruction of almost all intersecting texels. Texels only intersected by lines outside the diamond test area are not generated.*

For generating points for missed texels and detecting dispensable triangles, we again exploit the GPU rasterization rules of different primitives. We render the scene geometry in uv-space in the following order (see Figure 4): first, the vertices are rasterized as point primitive (left), second, the edges of the triangles are rasterized as line primitives (middle) and third, the geometry as triangles (right).

All problematic texels can be identified after readback to the CPU, as they have been rasterized by a point or line. The world-space position and normal for each of them is calculated, and recorded in a list P_{uv}. Additionally, we record a list of all the triangles T_{uv} that contributed to the uv-space rasterization, avoiding unnecessary micro triangles that do not create any texels.

During light accumulation, we can rasterize T_{uv} in uv-space and handle the problematic texels by subsequently rasterizing all points P_{uv}, thereby reconstructing all visible surface points. Figure 5 shows the result of the light accumulation (left) and the shaded scene geometry using a filtered light map including the detected missed texels in comparison to Figure 3.

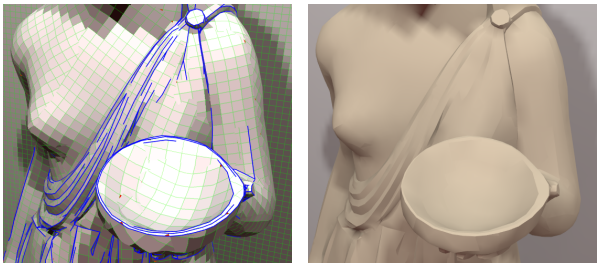


Figure 5: *Texel visualization of accumulated illumination using triangles T_{uv} and additional points P_{uv} (left). Scene geometry shaded by filtered light map (right).*

5 Virtual Light Creation

Real-time virtual light-based methods use a very small number of some hundreds of paths to distribute the light energy. Accurate global light distribution on the other hand requires the evaluation of tens or hundreds of thousands light paths to place VPLs. However, accumulating the contribution of all VPLs individually is not feasible. Since we are not constrained to real-time calculations, we can strive to create higher quality virtual lights that better capture the light distribution in a scene. The main idea in our approach is to cluster a larger number of VPLs into at set of higher order virtual lights. By introducing *virtual polygon lights*, we can efficiently evaluate the distributed light energy, while at the same time avoiding energy loss or spikes by improving energy transport accuracy with an analytical form factor evaluation.

5.1 VPL Generation

In order to pipeline the computation of VPL clusters on the CPU(s) and the subsequent computation of light maps on the GPU, we perform the simulation of each light source and each bounce separately, and thus generate the virtual lights one light bounce at a time. In addition to achieving nearly 100% utilization of the GPU for the shadow-map calculations of the virtual lights, the separation of bounces also leads to improved and automatically adapting clustering results for indirect illumination, incorporating the low spatial frequency attributes of higher diffuse bounces.

5.2 kd-Tree Clustering

In architectural scenes, a large number of surfaces are planar, allowing an easier approximation of light emission than on arbitrary surfaces. We exploit this fact in our approach by starting with a plane search in the VPL set. The search is performed using the RANSAC-based method introduced by Schnabel et al. [2007]: The VPLs are decomposed into connected planar subsets, each lying approximately on a plane, and a set of remaining VPLs, representing arbitrarily formed surfaces.

For further clustering of the segmented VPLs we use a hierarchical method with a kd-tree data structure that operates in 2D on detected planar segments and in 3D for the remaining VPLs: compute the axis-aligned bounding box of the VPLs, select the dimension with the greatest extent of this box, and split at the median position of this dimension. This operation is performed recursively on the subsets of VPLs until they are split into a set of clusters that contain approximately the same number of VPLs.

While this operation is very fast and creates clusters with approximately equal amount of light energy, forcing a median split does not cleanly separate lit areas split by shadow regions. In 3D there is the additional risk that the split plane coincides with a set of VPLs on a roughly planar surface, leading to a random separation and poor clustering. Therefore, we relax the requirement to split at the exact median p_m. If the VPLs are sorted according to their position along the split dimension, it is possible to linearly search for better split positions p_j along this dimension. We perform a search for large gaps in the VPL distribution along the split dimension, and choose the split point which optimizes a weighted criterion between gap size $|p_j - p_{j+1}|$ and index j relative to the median $I_m = \frac{n+1}{2}$:

$$\max_j \left\{ |p_j - p_{j+1}| \left(1 - \frac{2|I_m - j|}{n} \right)^{\rho}, j = 1 \dots n-1 \right\} \quad (1)$$

Our choice for the weighting parameter ρ is 0.5 to allow splits that are noticeably different from the median. This search significantly improves the clustering of the method while the computational complexity is still acceptable ($O(n \cdot \log n \cdot \log k)$, for n VPLs and k clusters).

For each cluster lying on a planar surface, we compute the 2D convex hull of all VPLs in that cluster. Due to limitations during the form factor evaluation on the GPU (max. 8 vertices), a simplified version of that hull polygon calculated using simple polyline vertex reduction forms the basis for computing the polygon of the virtual light. This simplified hull is extend by a border of size $1.5 \cdot \sqrt{Area_{hull}/n}$ and clipped against the kd-tree cell border in order to close the gaps between polygon lights in neighboring cells, assuming homogeneous photon density. The remaining VPL clusters, generated with the 3D kd-tree version, are approximated with sphere lights [Hašan et al. 2009].

The right image in Figure 6 demonstrates how the plane-aligned clusters and the emerging virtual polygon lights result in a close match to the geometry in the scene, leading to a reduction in illumination artifacts on corners prevalent in previous VPL-based approaches (see Section 7.2 for a detailed comparison).

6 Light Accumulation

First, we calculate direct lighting using the standard approach with shadow mapping into the light map. Of course, if the direct light is calculated dynamically, this step can be trivially omitted. For global illumination, the light map serves as render target and the irradiance of each virtual light is accumulated with additive blending. The pre-computed triangle list T_{uv} and point list P_{uv} (see Section 4) provide the surface points (p_y, n_y) of the scene at each light-map texel to the pixel shader. There, the irradiance $E_x(p_y, n_y)$ caused by a single virtual polygon light A_x can be calculated as:

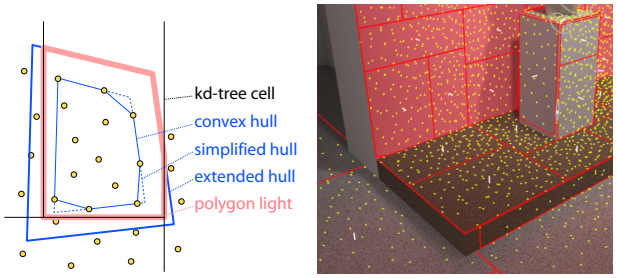


Figure 6: *Left: A polygon light is created out of a 2D VPL cluster by extending a simplified version of the convex hull and clipping it against the kd-tree boundaries. Right: Initial plane search leads to virtual polygon lights that closely match the underlying geometry, avoiding illumination artifacts in corner regions.*

$$E_x(p_y, n_y) = \Phi_{A_x} \cdot V_x(p_y) \cdot dF_{A_x \to dA_y}(p_y, n_y)/dA_y \quad (2)$$

The light power Φ_{A_x} is the sum of the powers of all VPLs on the polygon. Together with the analytic form factor $F_{A_x \to dA_y}$, which gives a precise scale for the light power at full visibility, the light energy is distributed accurately. We currently approximate the visibility term $V_x(p_y)$ of the polygon area using percentage closer soft shadows (PCSS) [Fernando 2005] with a cube shadow-map rendered from the polygon center.

6.1 Virtual Polygon Light Evaluation

Excluding visibility, the illumination transported by a virtual polygon light source with the area A_x with Lambertian emission characteristics to the differential area dA_y of a light map texel can be computed analytically using the polygon to differential area form factor, which can be derived analogously to the differential area to polygon form factor [Baum et al. 1989]:

$$dF_{A_x \to dA_y} = \frac{1}{2\pi \cdot A_x} \sum_{i=1}^{e} \vec{n_y} \cdot \frac{\vec{\Gamma}_i}{|\vec{\Gamma}_i|} \gamma_i dA_y \quad (3)$$

$$\vec{\Gamma}_i = \vec{V}_i \times \vec{V}_{i+1} \quad (4)$$

where e is the number of edges of the light source, n_y is the normal vector of the receiving differential area dA_y. $\vec{V}_i$ are the vectors from the differential area to the vertices of the polygonal light source, while γ_i denotes the angles between these vectors. A graphical representation of the form factor calculation is shown in Figure 7 (left). Note that although the equation is valid for non-convex polygonal light sources, our clustering method only creates convex polygonal lights.

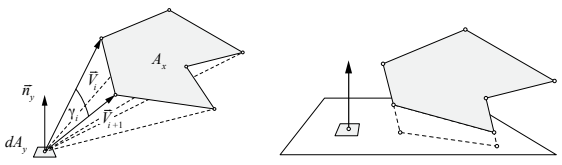


Figure 7: *Polygon to differential area form factor (left). Polygon light located partially below the plane of the receiving texel (right).*

Since we have an arbitrary configuration of polygon lights, they may be located partially below the plane of the receiving texel (see

Figure 7 (right)). In order to correctly compute the form factor, it is necessary to clip the polygon of the virtual light source against the plane of the receiving texel, as equation 3 is only valid for complete visibility.

The clipping operation and the polygon to differential area form factor computation are performed in the geometry and pixel shader stages of our light-map accumulation pass. The shader implementation performs early exit checks for lights with zero influence to (p_y, n_y) based on the orientation of the polygon. Additionally, a bias for points within an ϵ-region of 0.5 times the texel size of the plane the texel lies in is required to avoid the artifacts illustrated in Figure 8. For the form-factor calculation, the position p_y of points facing the polygon center is shifted along its surface defined by n_y to the intersection of the upper ϵ-border of the polygon plane. This prevents darkened edges in corners if a light-map texel position p_y falls within the ϵ-region. Due to numerical instabilities, it is necessary to treat polygon lights with extremely small solid angles separately in the form-factor calculations. In such a case, we switch to standard point-light evaluation, avoiding noise and speeding up the computation.

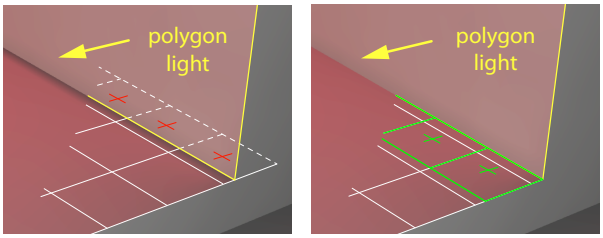


Figure 8: *Linear interpolation between lit and unlit light map texels causes shadow leaks (left). Moving texels within an ϵ-region into the illuminated area avoids these artifacts (right).*

7 Analysis and Results

In this section, we will first discuss the properties of our kd-tree clustering and general challenges of creating polygonal lights from VPL clusters. Second, we compare our virtual polygon lights to other higher order virtual lights used in recent publications. Finally, we give an overview of the performance of our complete light-map rendering system.

7.1 VPL Clustering

The kd-tree-based virtual light creation approach presented in Section 5.2 provides a fast and robust top-down clustering for several hundreds of thousands of VPLs. In comparison to k-means clustering as used in [Dong et al. 2009] and [Prutkin et al. 2012], our method guarantees an even global distribution. Dynamic updates and temporal coherence as required for real-time GPU implementations are not necessary for light-map computation. The clustering introduced in lightcuts [Walter et al. 2005] proceeds in bottom-up fashion and builds a complete hierarchy of all VPLs following an error metric, but it cannot be efficiently evaluated in a pixel-based selection on the GPU. Our kd-tree clustering builds local, geometry-aligned groups with an approximately equal amount of light energy. In order to create a suitable representation with *polygonal* virtual lights, the following difficulties had to be solved:

- alignment to geometry edges
- VPL density variations
- concave regions

The accurate energy transport provided by the polygon contour integral requires that polygonal lights are well aligned to the scene geometry in order to avoid bleeding artifacts. As already discussed in Section 5.2, initial plane clustering and simplifying the clustering to two dimensions results in much better alignment to the scene geometry. This was possible by using the fast shape detection of Schnabel et al. [Schnabel et al. 2007]. However, it has to be investigated if incorporating the actual scene geometry would open up additional possibilities to improve the virtual light alignment.

Since virtual polygon lights represent a surface area with constant diffuse emission, variations in VPL density cannot be accurately approximated. Clustering VPLs from different bounces or different light sources in a single step would lead to artifacts. Sparse VPLs from higher order bounces would disturb the clean shadow edges of the direct light, and lead to significantly reduced accuracy in the computation of the first indirect light bounce. Separating VPLs per light source and per bounce not only improves the utilization of CPU and GPU, it also avoids these problems, adapting to the properties of both the primary light as well as higher light bounces. Figure 9 shows the same scene rendered with a single clustered set of virtual lights and with separate clustering for each bounce.

A final problem are concave surface outlines formed by VPLs that are often created by shadows in the scene. Since only the convex hull of VPL clusters is approximated, the resulting polygonal lights might transport energy to hidden regions. We do not particularly address this problem, since by increasing the number of virtual lights, erroneous regions are successively split and removed.

7.2 Virtual Light Comparisons

As discussed by Hasan et al.[2009], using point lights leads to illumination spikes or energy loss if the illumination spikes are clamped. For this reason a number of different light models have been developed: disk lights (point, normal and area) with simple form factor approximation [Wallace et al. 1989] used in [Prutkin et al. 2012], sphere lights [Hašan et al. 2009], and VPL clusters [Dong et al. 2009] (these were called “virtual area lights” in the publication, but their evaluation is performed as a collection of individual point lights). In Figure 10, the virtual polygon lights are compared to these previous virtual light models.

The comparison illumination scenario consists of a single light bounce originating from a spot light. The ground truth image was rendered using a path tracer. In the comparison, all results were rendered using 197 virtual lights and evaluated in 5-times enhanced error images. Clustered lights use a total of 20k VPLs and have been created from the same clusters. The result of the original instant radiosity algorithm [Keller 1997] (column 1) clearly is the most biased, since the point lights miss exact information of local surfaces. Adding local cluster information, approximating an area and better orientation, the disk lights (column 2) already result in a good overall light transport. However, the point-based evaluation still results in dark corners. Sphere lights (column 3) have a different local evaluation yielding better results in corners, but tend to transport more erroneous light to their surroundings. By using all VPLs (with clustered visibility) to distribute the light energy, an overall accurate result is produced (column 4). The artifacts of darkened corners are smaller but very visible. Polygon lights (column 5) create a similar result and additionally resolve the artifact of missing light in corners. On curved surfaces, the accuracy is reduced to the polygonalization of the emitting geometry, but better than disk or sphere shapes due to their better adaptability to the surfaces.

This comparison shows that the most accurate diffuse virtual light based global illumination solution can be produced by choosing the type of virtual light depending on the property of the VPL cluster:

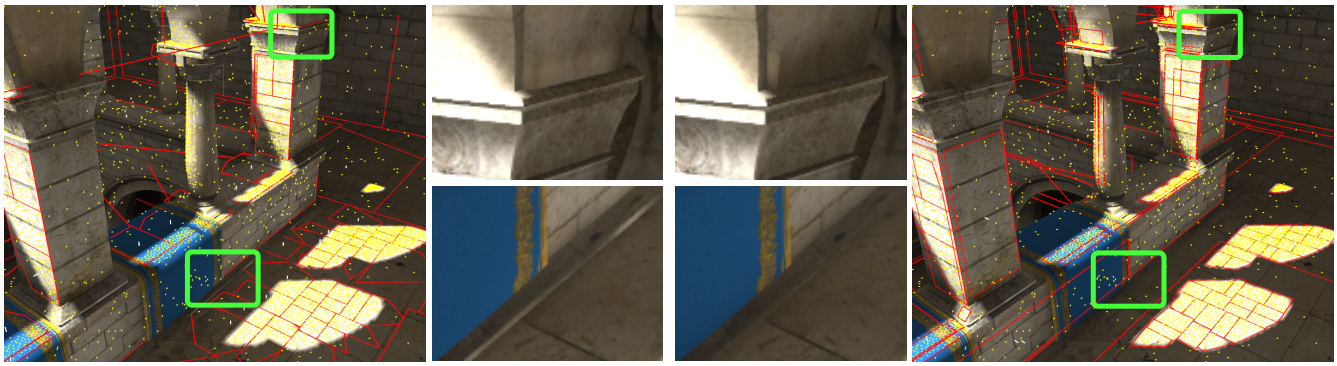


Figure 9: *The left images are generated from clustering all bounces together, for the right images each bounce is clustered separately. Note the difference in the virtual light sources for the lit areas created by the direct illumination through the arches, and the incorrect illumination in the left two close-ups.*

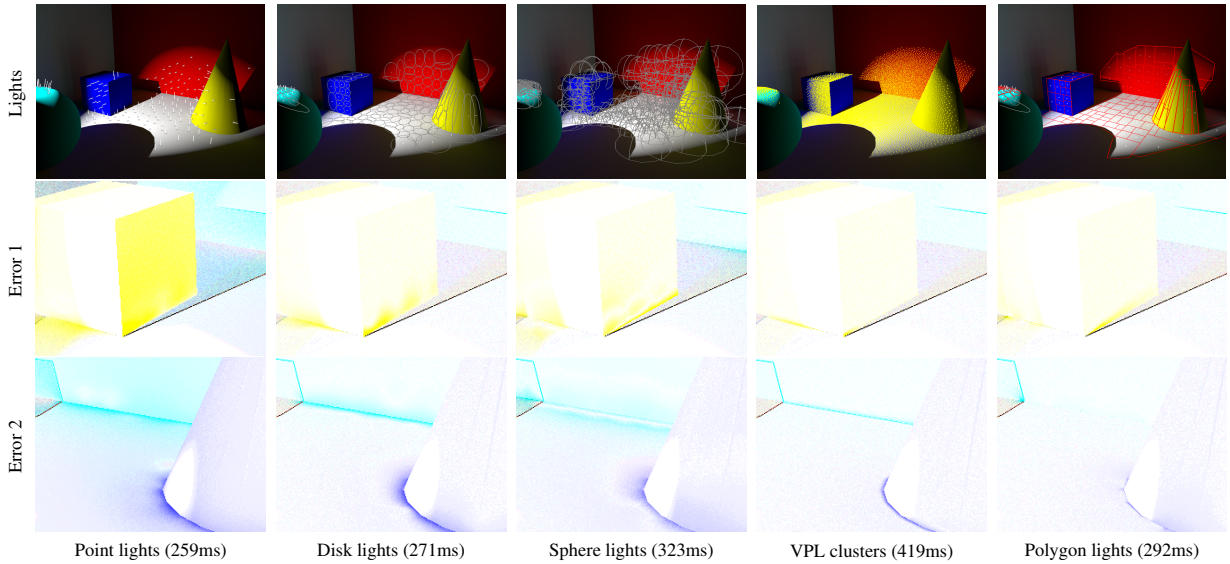


Figure 10: *Comparison of different virtual light types (197 lights/clusters created from 20,000 VPLs baked to a 3.2 MTexel light map): the top row visualizes the lights in the result, the two bottom rows show the 5-times enhanced inverse absolute error with respect to a path-tracing solution in two close-ups. Note that all point-evaluated virtual lights (Point Lights, Disk Lights, and VPL clusters) result in dark edges where two facing illuminated surfaces meet. Polygon lights strongly reduce this problem, but cannot completely avoid it for curved surfaces.*

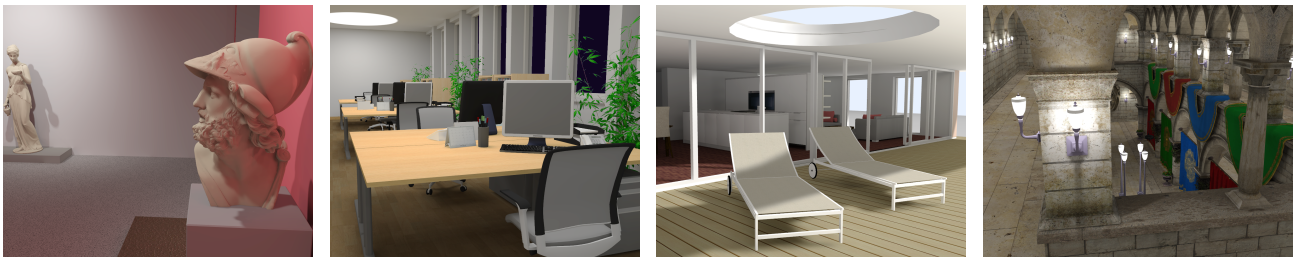
Polygons are to be preferred whenever an accurate outline of a planar region is given by a high VPL density. In planar regions with less density (e.g. higher bounces), where the risk of artifacts due to badly aligned polygon lights is increased, disk area lights can be used instead. Finally, clustered VPLs or sphere lights provide a good compromise to transport light of clusters with variations in VPL orientation and location.

7.3 Performance

We tested our approach on several scenes from different industrial fields as well as reference scenes. Figure 11 shows a museum scene (a) and an office (b), both modeled for industrial evaluation of different lighting scenarios, a penthouse model created for architectural planning and visualization purposes (c), and the Crytek Sponza Atrium, representing a part of a typical game level (d). These scenes differ significantly from each other: the Crytek Sponza Atrium is optimized for real-time applications, whereas the office and especially the penthouse scene are partly highly overtesselated and suffer from erroneous geometry. While this affects the calculation time due to the increased rendering effort, our approach still delivers a convincing lighting solution within an acceptable time frame.

Our application is implemented in C# and uses the DirectX 10 graphics API. The benchmark system is an Intel i7-920 processor and a NVIDIA Quadro 6000 graphics card with 6GB memory.

The pre-processing step described in Section 4 involves three rendering passes (points, lines and triangles), readback of the data and building the light-map geometry $T_{uv} + P_{uv}$. These tasks have to be performed once at loading time of the scene and when geometry is added during the interaction. The complexty of the light-map geometry strongly depends on the relation between geometry tessellation and the light-map resolution, while the processing time primarily scales with resolution. The following table shows the variety in different scenes:



(a) 8 sec / 2300 VLs (b) 24 sec / 2200 VLs (c) 30 sec / 3400 VLs (d) 15 sec / 1500 VLs

Figure 11: *Results generated with our approach: (a) a museum scene with high-frequency shadows and complex geometry, (b) an office illuminated by large area lights, (c) a penthouse demonstrating indirect light propagation and (d) the Sponza Atrium in a demanding illumination setting with 100 primary light sources.*

Scene	# triangles	# T_{uv}	# P_{uv}	time	resolution
Museum	213,572	36,646	75,786	0.64s	$2 \times$ 4k / 5.15MP
Sponza	279,163	179,328	319,538	1.15s	$4 \times$ 4k / 14.2MP
Office	716,960	123,795	226,647	0.84s	$2 \times$ 2k / 3.68MP

The number of additional points seems high, but in relation to the area covered by triangles, only about 5% of all texels are set by points. Since the rendering time is bound by the evaluation in the pixel shader, illumination information is created at equal costs per texel.

The time for the virtual light creation primarily depends on the number of initial VPLs. The graphs in Figure 12 show the measured total computation times with and without preliminary plane search. The time for the photon simulation to create the VPLs scales linearly at a rate of about 100k VPLs per second (included). The clustering is the most time-consuming task of the light creation and linearly scales with a fixed cluster size as well (in this range of VPLs, the logarithmic part of the algorithmic complexity does not yet come into play). The variations with enabled initial plane clustering are probably caused by scene-dependent detection of finer detailed planes.

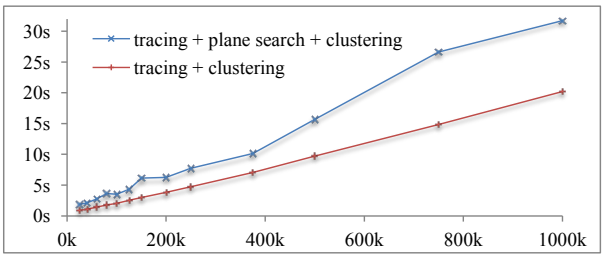


Figure 12: *Runtime of the light creation task dependent on the number of VPLs with 50 VPLs per cluster as target.*

The GPU-based light-map computation consists of two passes per virtual light. First, rendering a cube shadow map ($256^2 \times 6$) and second, accumulating the illumination by rendering the light-map geometry $T_{uv} + P_{uv}$. The PCSS visibility has been evaluated with 5 occluder search samples and 8 filter samples. Figure 13 shows the average time of accumulating a virtual light dependent on the light-map resolution. The timings were measured by averaging the computation time of 7k virtual lights. The ratio of polygon to sphere lights was approximately 1:1.

The accompanying video demonstrates that our system is well suited for interactive lighting design applications. The direct and indirect illumination is stored in a light-map, allowing arbitrary complex lighting scenarios. When objects or lights sources are moved, the direct illumination of affected lights is first subtracted from the light-map and then rendered in real-time during the interaction. The light map recalculation is scheduled interleaved with the scene rendering and blended with the old solution, continuously allowing interactions and illumination adjustments. Note that duplicating the light sources in the demonstration also invokes an update of the light-map layout including the rasterization test to rebuild the light-map geometry.

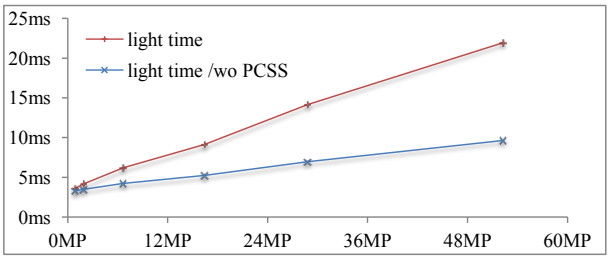


Figure 13: *Average virtual light evaluation time.*

8 Conclusion and Future Work

We have shown a fast method to pre-compute light maps for a complete scene using virtual polygon lights, avoiding long precomputation times. With consistent results in seconds and high-quality results in under a minute, we see the main application in lighting design for computer games and other light editing applications where fast turnarounds can drastically improve productivity, and where noise or flickering artifacts – often visible in walkthroughs using other interactive global illumination solutions – have to be avoided. Since it is light-map based, our solution offers high-quality and high frame-rate even on low-end devices.

Visibility computation is currently the most expensive task with potential for optimizations. Imperfect Shadow Maps [Ritschel et al. 2008] for example enable fast but crude visibility approximations. We would also like to evaluate the applicability of alternative soft shadow approximations such as Convolution Soft Shadows Maps [Annen et al. 2008].

Finally, it may be favorable to incorporate the idea of light cuts [Walter et al. 2005]: although individual light-source selection cannot be used for each light-map texel in our method, it could be applied in a large, tiled scene to reduce the number of virtual light sources that have to be evaluated, allowing the application of our method to extremely large scenes.

Acknowledgements

The competence center VRVis is funded by BMVIT, BMWFJ, and City of Vienna (ZIT) within the scope of COMET - Competence Centers for Excellent Technologies. The program COMET is managed by FFG.

References

ANNEN, T., DONG, Z., MERTENS, T., BEKAERT, P., SEIDEL, H.-P., AND KAUTZ, J. 2008. Real-time, All-Frequency Shadows in Dynamic Scenes. *ACM Trans. Graph. 27*, 3, 1–8.

BAUM, D. R., RUSHMEIER, H. E., AND WINGET, J. M. 1989. Improving Radiosity Solutions Through the Use of Analytically Determined Form-Factors. In *Proceedings of the 16th annual conference on Computer graphics and interactive techniques*, ACM, New York, USA, 325–334.

COHEN, M. F., WALLACE, J., AND HANRAHAN, P. 1993. *Radiosity and Realistic Image Synthesis*. Academic Press Professional, Inc., San Diego, CA, USA.

CRASSIN, C., NEYRET, F., SAINZ, M., GREEN, S., AND EISEMANN, E. 2011. Interactive Indirect Illumination using Voxel Cone Tracing. In *Symposium on Interactive 3D Graphics and Games*, ACM, New York, USA, 207–207.

DACHSBACHER, C., AND STAMMINGER, M. 2005. Reflective Shadow Maps. *Proceedings of the 2005 symposium on Interactive 3D graphics and games SI3D 05*, 203.

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*.

FABIANOWSKI, B., AND DINGLIANA, J. 2009. Interactive Global Photon Mapping. *Computer Graphics Forum 28*, 4, 1151–1159.

FERNANDO, R. 2005. Percentage-closer Soft Shadows. In *ACM SIGGRAPH 2005 Sketches*, ACM, New York, USA, 35.

HACHISUKA, T., AND JENSEN, H. W. 2010. Parallel Progressive Photon Mapping on GPUs. In *ACM SIGGRAPH ASIA 2010 Sketches*, ACM, New York, USA, 54:1–54:1.

HAŠAN, M., PELLACINI, F., AND BALA, K. 2007. Matrix Row-Column Sampling for the Many-Light Problem. *ACM Transactions on Graphics (TOG) 26*, 3, 26–es.

HASSELGREN, J., AKENINE-MÖLLER, T., AND OHLSSON, L. 2005. Conservative Rasterization. In *GPU Gems 2*, M. Pharr and R. Fernando, Eds. Addison-Wesley Professional.

HAŠAN, M., KŘIVÁNEK, J., WALTER, B., AND BALA, K. 2009. Virtual Spherical Lights for Many-Light Rendering of Glossy Scenes. In *ACM SIGGRAPH Asia 2009 Papers*, ACM, New York, USA, 143:1–143:6.

HERMES, J., HENRICH, N., GROSCH, T., AND MUELLER, S. 2010. Global Illumination using Parallel Global Ray Bundles. In *VMV 2010: Vision, Modeling and Visualization 2010*, 65–72.

JENSEN, H. 1996. Global Illumination using Photon Maps. *Rendering Techniques 96*, 21–30.

KAJIYA, J. T. 1986. The Rendering Equation. In *Computer Graphics (ACM SIGGRAPH '86 Proceedings)*, vol. 20, 143–150.

KELLER, A. 1997. Instant Radiosity. In *Proceedings of the 24th annual conference on Computer graphics and interactive techniques*, ACM Press/Addison-Wesley Publishing Co., New York, USA, 49–56.

LAFORTUNE, E. P., AND WILLEMS, Y. D. 1993. Bi-directional Path Tracing. In *Proceedings of Third International Conference on Computational Graphics and Visualization Techniques (Compugraphics '93)*, H. P. Santo, Ed., 145–153.

LAINE, S., SARANSAARI, H., KONTKANEN, J., LEHTINEN, J., AND AILA, T. 2007. Incremental Instant Radiosity for Real-Time Indirect Illumination. In *Proceedings of Eurographics Symposium on Rendering*, Citeseer, 277–286.

MCGUIRE, M., AND LUEBKE, D. 2009. Hardware-Accelerated Global Illumination by Image Space Photon Mapping. In *Proceedings of the 2009 ACM SIGGRAPH/EuroGraphics conference on High Performance Graphics*, ACM, New York, USA.

NOVÁK, J., ENGELHARDT, T., AND DACHSBACHER, C. 2011. Screen-space bias compensation for interactive high-quality global illumination with virtual point lights. In *Symposium on Interactive 3D Graphics and Games*, ACM, New York, USA.

PRUTKIN, R., KAPLANYAN, A., AND DACHSBACHER, C. 2012. Reflective Shadow Map Clustering for Real-Time Global Illumination. *Eurographics Short Papers*, 9–12.

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. In *ACM SIGGRAPH Asia 2008 Papers*, ACM, New York, USA.

RITSCHEL, T., ENGELHARDT, T., GROSCH, T., SEIDEL, H.-P., KAUTZ, J., AND DACHSBACHER, C. 2009. Micro-Rendering for Scalable, Parallel Final Gathering. In *ACM SIGGRAPH Asia 2009 Papers*, ACM, New York, USA, 132:1–132:8.

RITSCHEL, T., EISEMANN, E., HA, I., KIM, J. D., AND SEIDEL, H.-P. 2011. Making Imperfect Shadow Maps View-Adaptive: High-Quality Global Illumination in Large Dynamic Scenes. *Computer Graphics Forum (presented at EGSR 2011)*.

SCHNABEL, R., WAHL, R., AND KLEIN, R. 2007. Efficient RANSAC for Point-Cloud Shape Detection. *Computer Graphics Forum 26*, 2 (June), 214–226.

THIEDEMANN, S., HENRICH, N., GROSCH, T., AND MÜLLER, S. 2011. Voxel-based Global Illumination. In *Symposium on Interactive 3D Graphics and Games*, ACM, New York, USA, 103–110.

TOKUYOSHI, Y., SEKINE, T., AND OGAKI, S. 2011. Fast Global Illumination Baking via Ray-Bundles. In *SIGGRAPH Asia 2011 Sketches*, ACM, New York, USA, 25:1–25:2.

WALLACE, J. R., ELMQUIST, K. A., AND HAINES, E. A. 1989. A Ray Tracing Algorithm for Progressive Radiosity. *SIGGRAPH Comput. Graph. 23*, 3 (July), 315–324.

WALTER, B., FERNANDEZ, S., ARBREE, A., BALA, K., DONIKIAN, M., AND GREENBERG, D. P. 2005. Lightcuts: A Scalable Approach to Illumination. In *ACM SIGGRAPH 2005 Papers*, ACM, New York, USA, 1098–1107.

WANG, R., WANG, R., ZHOU, K., PAN, M., AND BAO, H. 2009. An Efficient GPU-based Approach for Interactive Global Illumination. In *ACM SIGGRAPH 2009 Papers*, ACM, New York, USA, 91:1–91:8.

WILLIAMS, L. 1978. Casting Curved Shadows on Curved Surfaces. *Computer Graphics (SIGGRAPH '78 Proceedings) 12*, 3 (Aug.), 270–274.