DOI: 10.1111/cgf.14053
Eurographics Symposium on Rendering 2020
C. Dachsbacher and M. Pharr
(Guest Editors)

Volume 39 (2020), Number 4

Adaptive Matrix Completion for Fast Visibility Computations with Many Lights Rendering

S. Wang[1] and N. Holzschuch[1]

[1] Univ. Grenoble-Alpes, Inria, CNRS, Grenoble INP, LJK

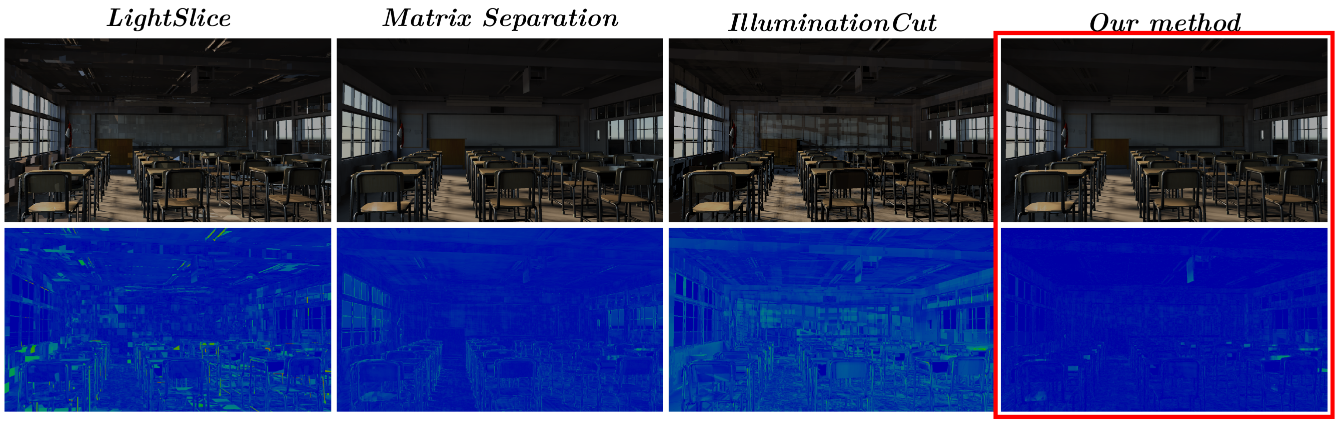

Figure 1: *Equal-time comparison between our method and other methods on the classroom scene.*

Abstract

Several fast global illumination algorithms rely on the Virtual Point Lights *framework. This framework separates illumination into two steps: first, propagate radiance in the scene and store it in virtual lights, then gather illumination from these virtual lights. To accelerate the second step, virtual lights and receiving points are grouped hierarchically, for example using Multi-Dimensional Lightcuts. Computing visibility between clusters of virtual lights and receiving points is a bottleneck. Separately,* matrix completion *algorithms reconstruct completely a low-rank matrix from an incomplete set of sampled elements. In this paper, we use adaptive matrix completion to approximate visibility information after an initial clustering step. We reconstruct visibility information using as little as 10 % to 20 % samples for most scenes, and combine it with shading information computed separately, in parallel on the GPU. Overall, our method computes global illumination 3 or more times faster than previous state-of-the-art methods.*

CCS Concepts
• ***Computing methodologies*** → ***Ray tracing;*** *Visibility;*

1. Introduction

Global illumination simulation lets us produce photorealistic rendering of virtual scenes. The process can be quite expensive, depending on the geometric complexity of the virtual scene. Several methods decouple the complexity of global illumination computation and the geometric complexity by using virtual light sources as an intermediate step: first, propagate illumination through the scene and store it as *Virtual Point Lights* (VPL). Then, compute illumination from these virtual point lights to the scene. To speed up the computations in the second pass, scalable Many-Lights methods approximate large numbers of light sources using just a few for some illumination computations. Examples include Lightcuts [WFA*05], Row-Column Matrix Sampling [HPB07], LightSlice [OP11]...

Visibility computations are still a bottleneck in the Virtual Point Lights methods: computing visibility from one virtual light source to a receiving point requires tracing a ray, and potentially testing the geometry of the entire scene for intersection. When we consider a set of light sources and a set of receiving points, the visibility information from each light source to each receiving point can be seen as a boolean matrix, with each coefficient equal to true if the corresponding elements are visible, and false otherwise.

Separately, several algorithms efficiently reconstruct a low rank matrix using a small set of samples: nuclear-norm minimization [CR09], alternating least squares minimization [HH09], and low-rank approximation methods [GTZ97]. First, they compute a small number of matrix coefficients, then they compute the full matrix that has these coefficients and minimizes certain criteria, such as the rank.

In this paper, we combine Adaptive Matrix Completion [KS14] with the Many-Lights method for faster global illumination computations: first, we compute virtual point lights and store them in a light tree. Then, we separate the points to be rendered into slices based on their position and orientation. We extract and render a set of clusters (a lightcut) from the light tree for each slice. To achieve this, we separate shading and visibility computations. Shading computations, computing the illumination from one virtual light to one receiving point, are run separately in parallel. For visibility computations, we first compute a small number of samples, then use a Matrix Completion algorithm to build the entire set of visibility samples. If we detect the number of visibility samples to be insufficient, we adaptively add more. Finally, we combine shading information with visibility information to produce the final image.

This separation between shading and visibility computations has several benefits: first, the visibility information matrix is boolean, simplifying several operations in the matrix completion algorithm and reducing computational complexity and memory usage. Second, computing the illumination without visibility can be run efficiently in parallel, for example on the GPU.

Our main contributions are:

- A new Many-Lights algorithm that separates visibility and shading,
- A variation of the Adaptive Matrix Completion algorithm for visibility,
- A method for importance sampling visibility in Many-Lights methods.

In the next section, we review previous works on both Many-Lights methods and Matrix Completion. We then briefly present the basis of Matrix Completion algorithms. In section 4, we present our algorithm for adaptive matrix completion for visibility. In section 5, we analyze our results and compare with existing algorithms. We conclude and present avenues of future work in section 6.

2. Previous works

2.1. Scalable Many-Lights Methods

Keller [Kel97] introduced Instant Radiosity, separating global illumination computations in two steps. In the first step, propagation, illumination is propagated through the scene and stored as Virtual Point Lights (VPL). These VPLs encode the outgoing illumination locally at multiple points in the scene, in a way that is independent from the mesh complexity. In the second step, gathering, we compute the illumination from each VPL to each pixel in the image.

To speed up computations, several methods group similar VPLs into clusters. Walter et al. [WFA*05; WABG06; WKB12] introduced Light Cuts: VPLs are organized into a hierarchy, and they select a sub-tree (a lightcut) for each receiver based on a refinement criterion. The idea was extended into Multi-Dimensional Light Cuts [WABG06] and Bi-Directional Light Cuts [WKB12], and by Bus et al. [BMB15b] into IlluminationCut. An interpolation method for these light-cuts was also investigated by Rehfeld and Dachsbacher [RD16], and a Virtual-Ray Light variant for participating media was proposed by Vibert et al. [VGS*19]. Bus et al. [BMB15a] investigate view-independant clustering using the concept of well-separated pair decomposition, allowing for faster rendering as clustering can be moved to pre-computation. Maria et al. [MMB*18] extend this method to take into account visibility.

Hasan et al. [HPB07] formulate the illumination problem as a matrix computation in Row-Column Matrix Sampling: each light source is a column, each pixel in the image is a row, and each element in the matrix contains the contribution from a given light source to a given pixel. They reconstruct the entire matrix using information extracted from a small number of rows and columns and clustering light sources based on their similarity. Ou and Pellacini [OP11] extended the approach in LightSlice, refining the clustering computed by grouping together similar surface samples.

Light Cuts-based algorithms are efficient in capturing local lighting information as they can generate clusters for an arbitrary number of receivers, but they are difficult to implement on the GPU, and clustering algorithms do not account for visibility. Matrix-based formulations such as Row-Column Matrix Sampling and Light Slice are easy to parallelize and take into account visibility, but have a much larger memory footprint when capturing local lighting features, and can show artefacts if this local information is not captured in fine enough details.

Huo et al. [HWJ*15] improve the matrix-based formulation using matrix separation to reduce the error. Huo et al. [HWH*16] used Adaptive Matrix Completion to compute illumination for participating media.

Our algorithm can be combined with existing Many-Lights algorithms, such as LightSlice. Our specificity is to separate visibility and shading computations, and to recover the visibility component using matrix completion.

2.2. Sparse Matrix Sampling in Rendering

Huang and Ramamoorthi [HR10] applied sparse sampling to Precomputed Radiance Transfer (PRT). They first select a small subset of vertices where they sample densely, then sample sparsely for the other vertices and interpolate from the dense vertices using locally low rank approximations. Wang et al. [WDT*09] used the Generalized Nyström method to reconstruct the light transport matrix from a relatively small number of acquired images for image-based relighting.

2.3. Visibility Approximation

Several methods exploit visibility coherence to reduce the cost of visibility computations, either in receiver space or in emitter space.

In *receiver* space, Hart et al. [HDG99] used flooding in image space to take advantage of the visibility coherence between neighbouring pixels. Fernandez et al. [FBG02] separated the scene into cells and only evaluate the light sources and occluders that are relevant for the current cell. Donikian et al. [DWB*06] progressively refine the average visibility from light sources to blocks of receivers until some variance threshold has been met. Vevoda et al. [VKK18] approximate visibility for light sampling by progressively refining a bayesian learning model with information from newly sampled lights.

In *emitter* space, Ritschel et al. [RGKM07; RGKS08] introduced coherent shadow maps, exploiting the angular coherence of shadows for an object by compressing all possible visibility queries into just a few shadow maps. This technique is per-object and does not scale well to complex scenes. Georgiev et al. [GKPS12] compute the contribution of a sparse set of lights in the scene to build a probability distribution for sampling lights.

Exploiting coherence in both emitters and receivers, Ben-Artzi et al. [BRA06] speed up visibility tests in scenes with environment maps by both discretizing the environment map and hierarchically sharing visibility between receivers, with the uncertain areas being evaluated using a flooding approach. Wu and Chuang [WC13] introduced VisibilityCluster, which approximates average visibility between light clusters and receiver slices for light sampling.

Our work also exploits the visibility coherence from both receivers and emitters. We exploit receiver coherence through slicing, and implicitly exploit emitter coherence with Adaptive Matrix Completion. Our importance sampling method also exploits coherence on both ends.

2.4. Matrix Completion

Matrix completion is a specific research topic in Computational Mathematics. The goal is to recover a matrix M by observing only a subset of its coefficients. Fazel [Faz02] showed that the problem is NP-hard. Candes and Recht [CR09] showed that it can be convexly relaxed as matrix nuclear-norm minimization, solved using convex methods. Cai et al. [CCS10] recover the matrix using a proximal gradient descent variant termed Singular Value Thresholding.

Nuclear-Norm minimization methods require computing the singular value decomposition (SVD) of the full matrix, which is computationally expensive for large dimensions. Haldar and Hernando [HH09], Wen et al. [WYZ12] and Tanner and Wei [TW16] used alternating least square methods for matrix completion. By factorizing the matrix $M^{m\times n} = X^{m\times k} Y^{k\times n}$, the rank receives an implicit upper bound k and eliminates the need for SVD. Unlike nuclear-norm minimization, this formulation is non-convex. However, it can be decomposed into separate convex sub-problems, allowing for the application of methods such as alternating direction method of multipliers.

Another approach is to project sub-sampled columns onto the range space of some small basis Q that spans the same space as

$$\begin{pmatrix} 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 \end{pmatrix} \qquad \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{pmatrix}$$

Figure 2: *The efficiency of matrix completion algorithms depend on matrix* rank *and* coherence. *These matrices are both of rank one; the matrix on the right is very coherent, and will require more samples to accurately complete.*

the full matrix, using $QQ^\dagger$, where $Q^\dagger$ is the pseudo-inverse of Q. If the sub-sampled column already lies within Q, the full column can be reconstructed accurately with the obtained coefficients, and if not, a low-rank approximation of the column is obtained. The Generalized Nyström method [GTZ97] reconstructs a full matrix from a set of sampled rows and columns. A major issue with this method is that the intersection of the rows and columns has to have the same range space as M for the completion to be accurate. Krishnamurthy and Singh [KS14] partially alleviate this concern with Adaptive Matrix Completion, which extends the idea by conducting the column sampling in an adaptive and progressive manner.

For efficient computation in the Many-Lights Method, we need the matrix completion algorithm to be faster than directly evaluating the entries in the visibility matrix. We found the Adaptive Matrix Completion algorithm [KS14] to be particularly suited for this as it is both fast and can be modified to scale well when dealing with boolean data, such as visibility. The progressive nature of the algorithm also allows us to derive an importance sampling strategy for visibility.

3. Background

3.1. Matrix Completion

Matrix completion is a specific research problem in Computational mathematics: Trying to infer a complete matrix, knowing only a small subset of its coefficients. In general, the problem is ill-posed and has an infinite number of solutions, so we add a constraint on the matrix rank: Finding the matrix that has the smallest possible rank and whose coefficients are equal to the known coefficients:

$$\min_X \operatorname{rank}(X) \quad \text{s.t.} \quad A(X) = b \tag{1}$$

where $X \in \mathbb{R}^{m\times n}$ is the matrix to be recovered and $A : \mathbb{R}^{m\times n} \to \mathbb{R}^p$ is a linear map that maps X to the observation vector $b \in \mathbb{R}^p$.

In order to complete the matrix accurately, we assume the partial observations b fully represent the latent factors of the matrix. Two matrix properties determine how many observations are required:

- The matrix **rank**, which indicates the number of latent factors of the underlying data. As the matrix rank increases, so does the number of required observations.
- The matrix **coherence**: defined as how closely the singular vectors correlate with the standard basis, and can be thought of how *localized* the features of the matrix are. This impacts the required

Algorithm 1 The Adaptive Matrix Completion algorithm

```
procedure COMPLETE($M^{r\times c}, k$)
    $Q \leftarrow \emptyset$
    Let $\Omega$ be the set of $k$ indices drawn from $U(0,r)$
    for Each column $M_i \in M$ do
        Sample $M_{i_\Omega}$
        if $\left\|Q_\Omega Q_\Omega^\dagger M_{i_\Omega} - M_{i_\Omega}\right\|_2^2 > 0$ then
            Fully sample $M_i$ and add to $Q$
            Resample $\Omega$ from $U(0,r)$
        else
            $M_i \leftarrow Q Q_\Omega^\dagger M_{i_\Omega}$
        end if
    end for
    return $M$
end procedure
```

number of observations: more localized features require a higher number of samples. See Figure 2: Both matrices are rank one, but the leftmost matrix can be completed with a single sample, while the rightmost matrix has a very localized feature and will require sampling exactly on this feature.

3.2. Adaptive Matrix Completion

The Adaptive Matrix Completion (AMC) algorithm is a specific matrix completion algorithm. It completes matrices by iteratively adding columns. We start with an empty matrix M that we need to complete, and an empty basis Q. We maintain Q to be a full-rank set of basis vectors spanning the same range space as M. We iteratively attempt to add new subsampled columns M_{i_Ω} to the matrix by attempting to reconstruct the full vector using:

$$M_i = Q Q_\Omega^\dagger M_{i_\Omega} \tag{2}$$

where $A^\dagger$ is the pseudo-inverse of A. This orthogonally projects M_{i_Ω} onto the range space of Q_Ω, and fills in the rest of the vector with the obtained coefficients. The reconstruction error E is defined as:

$$E = \left\|Q_\Omega Q_\Omega^\dagger M_{i_\Omega} - M_{i_\Omega}\right\|_2^2 \tag{3}$$

If $E > 0$ then M_{i_Ω} does not lie within the range space of Q_Ω, and thus by proxy M_i does not fall inside the range space of Q. In this case, we fully sample M_i and add it to Q, expanding its range space. If there is no reconstruction error ($E = 0$) then it is likely that M_i has been accurately constructed, assuming enough samples were taken. The full algorithm is detailed in Algorithm 1.

This algorithm has several advantages for us:

- The adaptive column sampling relaxes the constraints on column coherency, requiring less samples to accurately complete the matrix.
- As columns are added iteratively, it is possible to introduce an importance sampling strategy that takes advantage of already computed information.
- As the algorithm relies on projection to range spaces, we can take advantage of the limited range space of boolean visibility matrices to avoid computing Equation 2, allowing for better scalability.

3.3. The Many-Lights Matrix

One method to represent the Many-Lights problem is to use a matrix, with one column for each light source and one row for each receiver. Each matrix element represents the contribution from this specific light source to this receiver. From this matrix M, we compute the final colour at each receiver r_i as the sum of all elements for the row i. A key property of this matrix is that it is approximately low rank [HPB07]. We can approximate it with just a limited number of samples using matrix completion methods.

4. Method

4.1. Overview

Our goal is to speed up Many-Lights rendering while retaining as much accuracy as possible. To achieve this, our approach is divided into the following main steps:

- generate receiver points and split them into slices
- generate VPLs and cluster them for each slice
- apply AMC to approximate each slice's visibility information
- shade the receivers with the approximated the visibility information and combine all the shading information to generate the final image

4.2. Generating and slicing receivers

We generate receivers by casting rays from the sensor into the scene, and tracing them until they encounter a non-specular surface. These receiver points, using a 6-dimensional vector to represent their position and weighted normal, are then partitioned into slices using a kd-tree by iteratively splitting the nodes of the tree by the largest dimension until the number of receivers within a node is less than a predetermined number. We use $0.05 * D$, where D is the scene bounding sphere radius, for the normal weight, and 1024 for the slice size threshold.

Slicing the receivers according to their position lowers the rank of the matrix by disregarding more distant, and likely irrelevant, information. This allows for both a lower number of required samples to accurately construct the matrix, and a way to adapt sample rates across the scene depending on how high rank a slice is. Incorporating orientational information into the slicing aids the clustering, as fewer light clusters are required to accurately approximate the local lighting information.

4.3. VPL generation and clustering

We generate our VPLs similar to how photons are generated in the photon mapping method [Jen01]. We sample a set of random walks of a specified maximum length from randomly sampled light positions, and record a VPL at each vertex. Here, the direction of the segments of a random walk depend on the BSDF of the previous vertex. Walks can be terminated early based on russian roulette.

To reduce the problem space, we cluster the lights for each slice.

For this, we use a modified version of Lightcuts, which achieves a good balance between computation time and cluster quality.

Unlike the original method, we aim to extract lightcuts for slices of receivers rather than single points. We do this by simply refining according to the maximum estimated error across all points in the slice. This modification is required as the receivers in the matrix need to share the same set of lights. It also provides a significant decrease in computation time, as far fewer lightcuts are computed.

We also change the construction algorithm of the light tree from bottom-up to top-down; the lights are iteratively separated based on their projection on the largest principal axis formed from their positional and scaled normal values. This change was necessary due to the $O(n^2)$ construction complexity of the light-tree, which leads to prohibitively long construction times for scenes with large numbers of VPLs.

We could also use LightSlice for our clustering step; in our experiments, it generally performs worse in equal time computations and it has a large memory footprint, making it infeasible with large number of VPLs.

4.4. The Visibility Matrix

One issue with using matrix completion methods on the Many-Lights matrix is that it is difficult to determine how many initial samples are required. This is because both the rank and coherence of the matrix depend on the material and geometric complexity of the scene. We alleviate this problem by splitting the matrix into shading and visibility components:

$$M^{r\times c} = S^{r\times c} \bullet V^{r\times c} \qquad (4)$$

where $\bullet$ is the element-wise multiplication of the two matrices, V is a matrix with boolean coefficients that specifies whether or not there is a direct line of sight between the light source and the receiving point, and S is a matrix with real coefficients encoding all other shading information between the receiving point and light source.

Our aim is to use matrix completion for V and to compute S directly on the GPU. This improves the matrix completion process as:

- completing V is more robust due to the removal of material properties and incident energy.
- Since V is boolean, we reduce computational complexity and memory usage (section 3.2).
- It is easier to importance sample V with Adaptive Matrix Completion, alleviating the matrix coherence issues (section 4.5.2).

4.5. Adaptive Matrix Completion

We define a visibility matrix for each slice, based on their receivers and clustered VPLs. These matrices are then completed using AMC with the following modifications specially tailored to boolean visibility matrices

- we change the projection technique to a boolean match between columns of Q_Ω and M_{i_Ω}
- we use importance sampling for the rows (instead of uniform)
- we dynamically adjust the per-slice row sample-rates

The full algorithm is detailed in Algorithm 2.

Algorithm 2 Boolean Adaptive Matrix Completion

```
procedure COMPLETE($M^{r\times c}, \alpha, \beta, \omega$)
    $Q \leftarrow \emptyset$
    $k \leftarrow ceil(\alpha \times c)$
    $d_i^k = \frac{1}{k} \forall 0 \le i < k$
    for Each column $M_i \in M$ do
        Sample $\Omega$ as $\alpha$ indices from discrete distribution $||d||$
        Sample $M_{i_\Omega}$
        $P = \left\{ x | x \in Q \cap (x_\Omega = M_{i_\Omega} \cup !x = M_{i_\Omega}) \right\}$
        if $P = \emptyset$ then
            Fully sample $M_i$ and add to $Q$
            $d \leftarrow Update(d, M_i)$
        else
            $\overline{M_i} \leftarrow P_j, P \in \mathbb{R}^{q\times n}, j = U(0,n)$
            if $Verify(\overline{M_i}, \Omega, \beta)$ then
                $M_i \leftarrow \overline{M_i}$
            else
                Fully sample $M_i$ and add to $Q$
                $d \leftarrow Update(d, M_i)$
                $\alpha \leftarrow Max(\alpha + \beta, \omega)$
            end if
        end if
    end for
    return $M$
end procedure
```

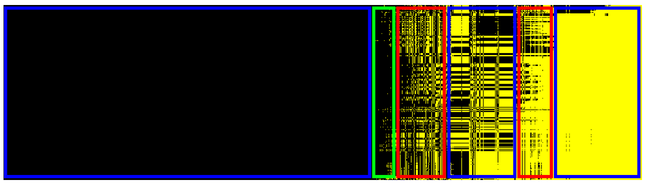
Figure 3: *A sorted lower rank visibility matrix of a slice within the kitchen scene with the VPLs as columns and receivers as rows. Low-rank columns are marked in blue, high-rank columns are marked in red, and highly coherent columns are marked in green.*

4.5.1. Eliminating the pseudo-inverse

Visibility is boolean and the low-rank nature of V is caused by large coherent areas in the matrix (Figure 3). This allows us to replace Equation 2 with merely verifying if either M_{i_Ω} exists in Q_Ω or $!Q_\Omega$, where $!A$ is the boolean not of A. If some matching column Q_{Ω_k} or $!Q_{\Omega_k}$ is found, then it is assigned to M_i. In the case where there are multiple matching columns, it means that there are multiple possible solutions. We resolve this by selecting a randomly sampled column. However, this does mean that the sampling quality may be insufficient. We use adaptive sample-rate adjustment and importance sampling to mitigate this.

This algorithm is linear in complexity with both the rows and columns of Q, instead of the quadratic complexity. The matching process can be reduced to basic boolean operations, further lowering both the memory and processing footprint. A drawback to this method is that it may be insufficient for general boolean matrices

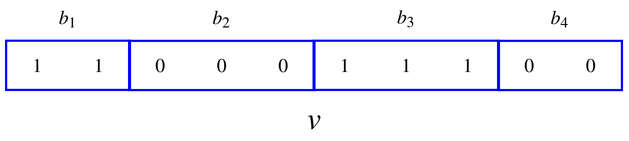

Figure 4: *The visibility vector v is separated into four buckets based on equal contiguous values*

Algorithm 3 Update step for importance sampling

```
procedure UPDATE($d, c$)
    Let $b^n$ be a vector of buckets of $c$ containing indices
    $d_i \leftarrow d_i + \frac{1}{size(b_j) \times n}, i \in b_j$
    return $d$
end procedure
```

where more than one column of Q needs to be considered to reconstruct M_i.

4.5.2. Importance Sampling the rows

We want to allocate samples proportionally to the features in the vector, rather than using uniform sampling. In our case, this means areas where visibility changes a lot. We model this by dividing the vector into buckets based on contiguous equal values, with each bucket having an equal probability of being selected, and each index within the bucket also having an equal selection probability (Figure 4). With this, the probability of selecting an index can be computed as:

$$P(v_j) = P(B)P(v_j|B) = \frac{1}{b \times s} \tag{5}$$

where b is the number of buckets, j is the index of the vector to be sampled, B is the bucket that the index belongs to, and s is the number of indices in the bucket.

As it is difficult to know the distribution of buckets beforehand, we exploit the iterative column sampling of AMC by using previously sampled information to build our distribution. To ensure that there is some level of similarity between the columns, we further slice the matrix column-wise based on the quadrant of the incoming light direction to the centroid of the slice. These sliced matrices are recovered separately and then combined to form the actual visibility matrix for the slice.

For each sliced matrix $V_i^{m \times n}$, we maintain a discrete distribution d_i^m which we use to sample the columns. This distribution starts with a uniform probability of $\frac{1}{m}$, and is updated every time a column is fully sampled using

$$d_{i_j}^{\text{new}} = d_{i_j}^{\text{old}} + P(v_j) \tag{6}$$

As d_i is not normalized, we instead sample from $||d_i||$. Algorithm 3 details the update process.

4.5.3. Dynamically adjusting per slice row sample-rate

The original AMC algorithm uses a constant row sample-rate. This is sub-optimal for completing the visibility matrix as the rank and coherence of slices are not uniform, and have different ideal sample-rates. Thus, we aim to adopt a method that dynamically adjusts the row sample-rate of the slice.

Ideally one would adjust the sample-rate according to both the slice rank and coherence, but these can't be determined without sampling. We therefore use a verification approach, where we evaluate the correctness of each completed column with a set of sparse samples, and increase the sample-rate accordingly up until a maximum sample-rate if the column is found to be incorrect. The sparse sampling and verification procedure is detailed in Algorithm 4.

Algorithm 4 Verification for dynamically adjusting row sample-rates

```
procedure VERIFY($c^n, \Omega, \beta$)
    $k \leftarrow ceil(\beta \times n)$
    Let $\Pi$ be the set of $k$ indices drawn from $U(0,n), \Pi \cap \Omega = \emptyset$
    Sample $\overline{c_\Pi}$
    if $\overline{c_\Pi} \neq c_\Pi$ then return false
    elsereturn true
    end if
end procedure
```

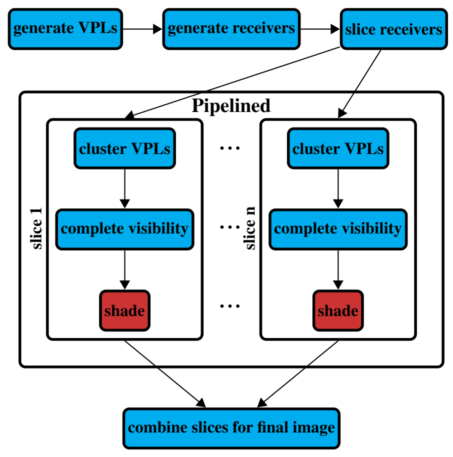

Figure 5: *The architecture for our system. Nodes in blue are computed on the CPU, whereas nodes in red are computed on the GPU.*

4.6. Shading

The shading of receivers is performed on the GPU as it only requires local information. We use the Virtual Spherical Lights [HKWB09] method in glossier scenes, and oriented diffuse VPLs with clamping for more diffuse scenes.

4.7. Summary

The full rendering algorithm can be summed up into the VPL propagation, receiver generation and slicing, clustering, visibility completion, and the shading and rendering steps (Figure 5). The last three steps are pipelined so that the CPU can process new slices while the GPU is busy with others.

5. Results and Discussion

5.1. Experimental setup

We evaluate our method across 12 different scenes, selected to cover a diverse range of possible scenarios, against ground truth images obtained by brute-force computing all the VPL contributions for the scenes. We only show comparisons for the classroom scene in this paper due to space limitations, but full information and results of all our test scenes can be found in the supplemental. Each scene is rendered with 100k VPLs at 2×2 samples per pixel. We treat each sample as a separate receiver with its own shading and visibility, and average the values for the final pixel colour. These scenes were obtained from Bitterli [Bit16] and McGuire [McG17].

Our method is adaptive by nature; we did not need to tune the parameters. We use $\alpha = 2.5\%, \beta = 2.5\%, \omega = 40\%$ for all test scenes. To achieve a time-error tradeoff, we set an exact number of clusters to be generated per slice.

We compare our method to LightSlice, IlluminationCut, and Matrix Separation. We also compare our method to just using the modified Lightcuts algorithm with full visibility samples to gives an idea of the performance gain achieved from completing visibility. We obtained the source code for Lightslice [OP] and IlluminationCut [BMB] online, and the Matrix Separation source code directly from the authors. We ported these integrators into our system, built in Mitsuba [Jak10], to have matching material models. We obtain a time-error tradeoff of these techniques by varying either the number of clusters per slice or the error threshold. We use the root-mean-squared error (RMSE) to the ground-truths as our error metric, and the total processing time as our time metric. We don't compare IlluminationCut or Matrix Separation in our four glossier scenes, the kitchen, the bathroom, modern hall, and grey & white room, as they don't handle Virtual Spherical Lights (VSLs).

We performed our tests on an Intel Xeon ES-2630 v3 @ 2.40GHz CPU with a nVidia GeForce GTX 1080 Ti and 32GB of RAM. We found that for the largest scenes, such as San-Miguel, our memory usage topped at about 10GB. This is due to the batching size of our GPU jobs as well as the number of slices being processed in parallel. We found each slice to use roughly 70MB in the absolute worst case.

5.2. Test scenes

Figure 6 shows images generated with our method for four different test scenes. Our method works best with the Sponza attrium, using only 7.5 % of the visibility samples. The worst test scene is the hairball scene, with 37.98 % visibility samples. We used around 15 % visibility samples for the San Miguel and kitchen scenes. Rendering times range from 32.6 s (Sponza) to 114.15 s (kitchen). Our method struggles with the hairball as there are a large number of high frequency occlusions, resulting in near full visibility sampling of many receiving points; even in this worst case, our method still outperforms the other methods.

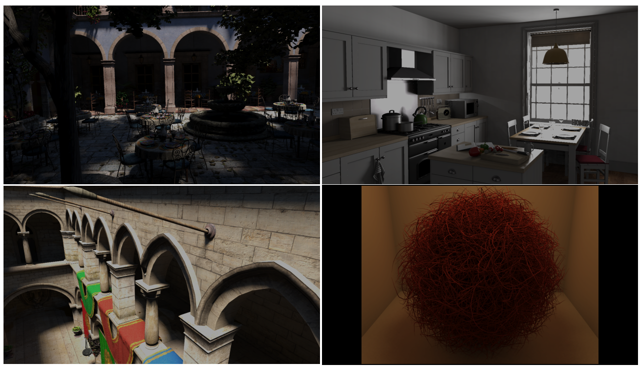
Figure 6: *Images generated by our method across scenes of varying complexity at* 2×2 *samples per pixel at 100k VPLs. All images are* 1280×720*, except for the hairball which is* 800×600.

5.3. Comparisons

Figure 6 shows equal time comparisons and a time versus RMSE plot for the classroom scene. Similar comparisons for the other scenes are provided in the supplemental.

5.3.1. LightSlice

Compared to LightSlice, our method generates images with similar or better quality at least 3 times faster. An exception to this is in glossy scenes, where the methods are comparable. However, this is only average error and LightSlice has many slice-based artefacts. LightSlice also does not scale well to number of VPLs, as the initial clustering matrix would become much larger, impacting memory and processing. This can be counteracted with larger slices, but this comes at the cost of accuracy and more slice-based errors.

5.3.2. Matrix Separation

Matrix Separation shows similar initial performance to our method in many scenes. However, it converges quickly to a suboptimal image, whereas our method continues improving with longer processing times. An explanation for this is its predictor validatation procedure. Raising the error threshold should provide better results, but the samples are few and are performed uniformly, easily missing important features. Its clustering method, which borrows information from nearest neighbours, with some sparse visibility samples, can also be a cause. If the sparse visibility samples fail to properly capture the visibility of the neighbouring tree nodes, one could end up with premature termination of the refinement process. This method also does not translate to scenes with high frequency effects, such as the hairball. This is because these features can come up as error during matrix separation, and are subsequently removed, causing a blurring effect. We also had to set the column energy threshold to be very low in order to avoid predictor visual artefacts. This resulted in very high visibility sample rates for some scenes.

5.3.3. IlluminationCut

IlluminationCut is more than 10 times slower than our method in all scenes. This can be due to excessive refinement in areas with a

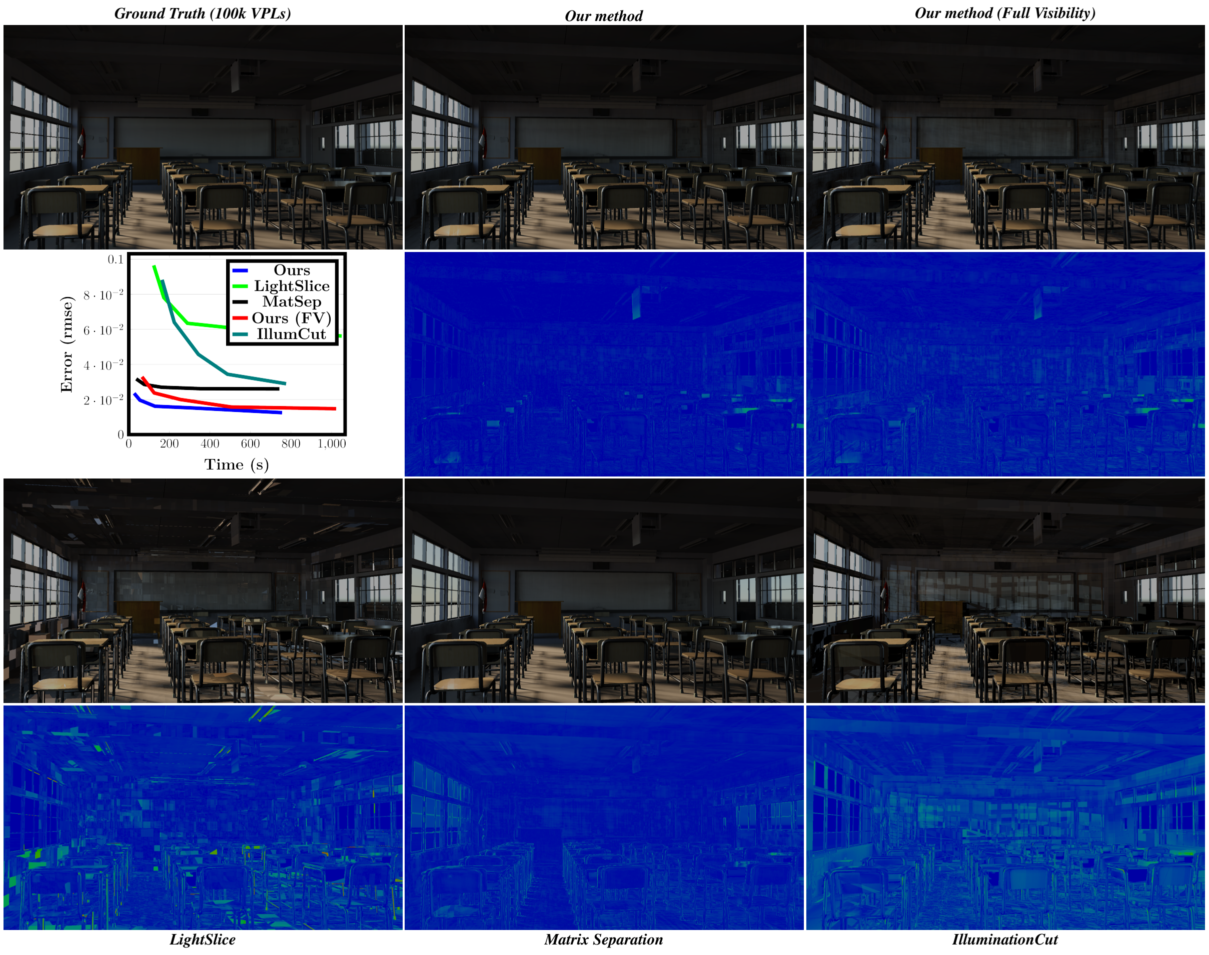

Figure 7: *Equal time images (~55s) and time error plot generated classroom scene. Error images provided show l_1 error.*

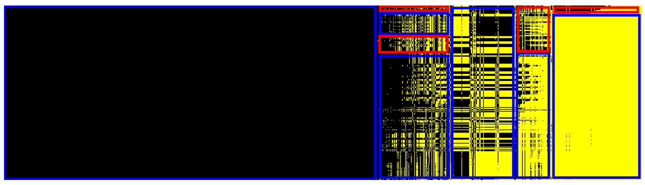

Figure 8: *A more ideal way of completing the matrix. We ideally want to mostly fully sample the areas of the matrix outlined in red, and complete the areas outlined in blue with as few samples as possible. Currently, the small regions outlined in red are causing the entire columns to be sampled.*

low error upper bound, and to sub-optimal splitting where the light tree splits far more than the gather tree. Although we also use a similar clustering method, we don't use a gather tree, and instead cluster based on pre-determined slices instead, drastically reducing the number of lights for each receiver. Although this is simpler and produces potentially worse clusters, the performance gain compensates. We also found that their adaptive visibility sampling strategy outperforms ours in simple scenes. There are two reasons for this: they do not allocate many samples, and miss certain visibility features within the matrix; they allocate far fewer samples to featureless areas. This is a drawback of our method, as we fully sample a column even if it is slightly different from the existing ones. We would ideally either only fully sample the piece of the column that is different and complete the rest (Figure 8). Our lower-bound to the number of samples taken per column is also sub-optimal, as we can oversample featureless columns.

5.3.4. Full visibility sampling

Compared to the modified LightCuts with full visibility sampling, our Adaptive Matrix Completion for visibility is 3 to 10 times faster. An exception to this is the hairball scene where we have to take a large number of visibility samples, where it is only 2 times faster. This speedup is expected, as the completion process has very little overhead, making the performance gain from omitting casting visibility rays much more noticeable. Our method also has similar error levels to Lightcuts with full visibility when both methods use the same number of clusters (see supplemental).

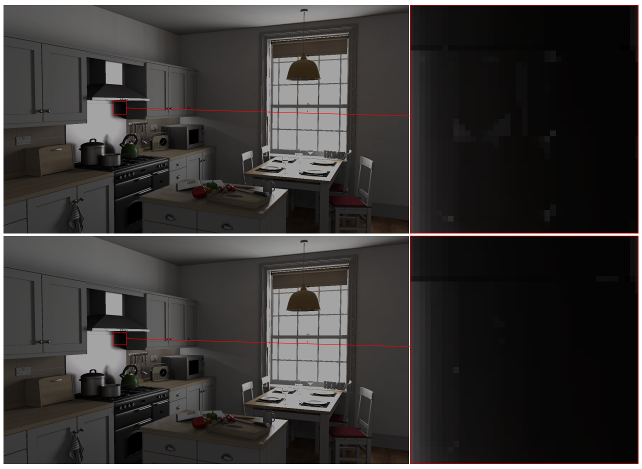

Figure 9: *Light leaks can be caused by insufficient sampling in high rank areas (top). Increasing the number of verification samples improves this (bottom).*

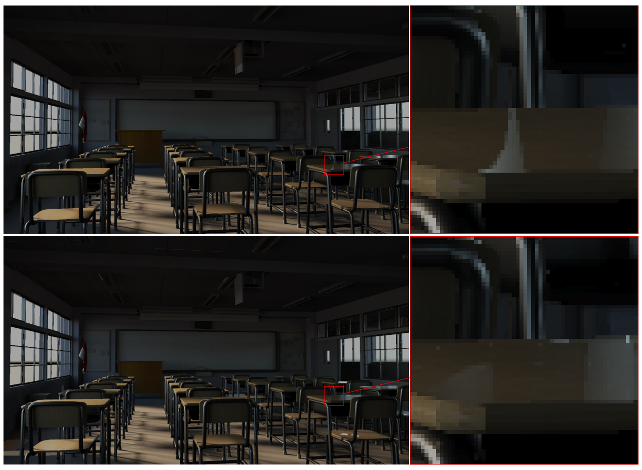

Figure 10: *Poor refining caused by clustering not handling visibility can be enhanced by glossy surfaces, causing banding (top). Switching to a LightSlice alleviates this (bottom).*

5.4. Artefacts

Our method produces two types of artefacts. Figure 9 shows visibility artefacts that can occur in high rank regions, manifesting in the form of either light leaks or darkened patches. These are caused by not allocating enough samples, and can be fixed by increasing the verification sample percentage. This comes at a cost as more verification samples are used across the entire image. Figure 10 shows bright banding on glossy surfaces. This is caused by our clustering method, which does not account for visibility. Refining becomes sub-optimal in scenes where large numbers of lights are occluded, and glossy surfaces greatly magnify this. This can be remedied by switching to a clustering algorithm that accounts for visibility.

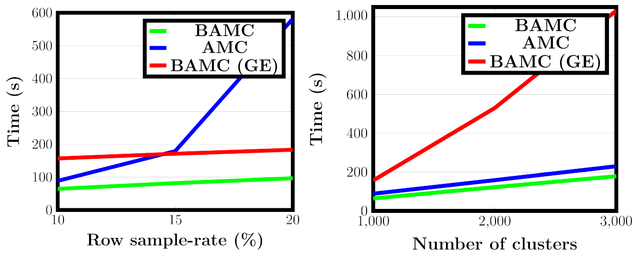

Figure 11: *Scaling of boolean AMC both to row sample-rate (left) and number of clusters (right) compared to using standard AMC and Boolean AMC with Gaussian elimination.*

5.5. Using other clustering methods

Although we use a modified version of Lightcuts, our algorithm should theoretically be independent of clustering methods as long as they extract sets of clusters for groups of receiving points. This also means that it can also be used when no clustering is performed. However, we found it to not work well with LightSlice in complex scenes. The reason for this is twofold: LightSlice has larger slices, making the full column sampling far less efficient. Reducing the slice size is possible, and is what we did for Figure 10, but drastically increases the processing time and memory usage; LightSlice already accounts for visibility, and thus the gain from matrix completion is less obvious. Improving our completion strategy should allow our method to work with larger slices. Images rendered using brute force and LightSlice are provided in the supplemental.

5.6. Boolean Adaptive Matrix Completion

To validate our boolean matching method, we compare both its accuracy and computational speed to both the pseudo-inverse method used in the original AMC algorithm, and Gauss-Jordan elimination for general boolean matrices.

The general idea behind the Gauss-Jordan method is to reduce Q^T to reduced row echelon form $\overline{Q^T}$ using Gauss-Jordan elimination, and then finding the rows of $\overline{Q}$ that can reconstruct M_i^T by checking the sampled values corresponding to the indices of the leading 1s of each basis vector. More information on this method can be found in the supplemental.

Figure 11 shows how AMC with boolean matching scales both row and column-wise compared to standard AMC and boolean Gaussian elimination for the San Miguel scene. Performance increases are similar across all scenes, with the supplemental containing performance statistics across three different scenes. While the computation times are not too different for low row sample-rates and clusters, original AMC and Gaussian elimination quickly becomes computationally infeasible when row sample-rate and number of clusters are increased respectively.

The main reason for the poor scaling of standard AMC (Equation 2) is because it requires SVD, which has a complexity $O(mn^2)$, where n is the smaller dimension. Since that is normally the rows for our matrices, the algorithm scales quadratically in that direction. Gaussian elimination for non-square matrices is likewise $O(mn^2)$.

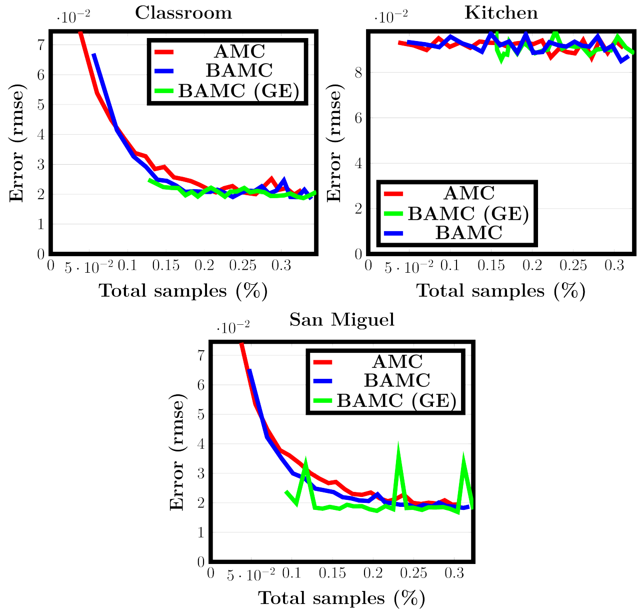

Figure 12: *Total samples vs RMSE for BAMC, BAMC with gaussian elimination, and AMC. It can be seen that the three methods have comparable accuracy.*

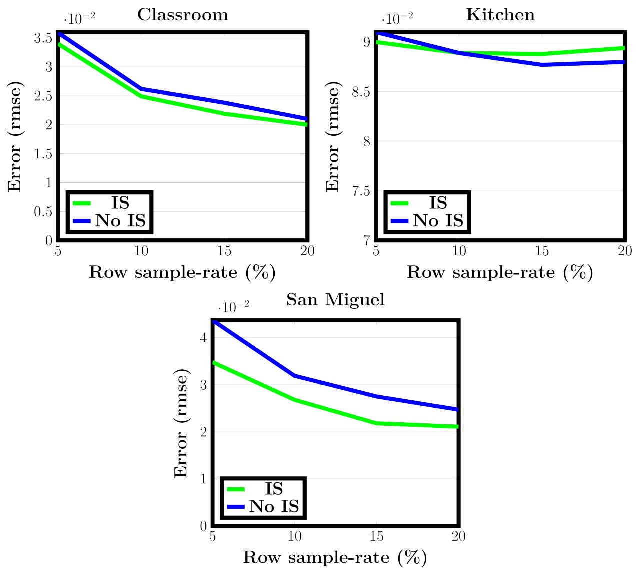

Figure 13: *Samplerate versus error plotted compared for uniform and importance sampling across three scenes.*

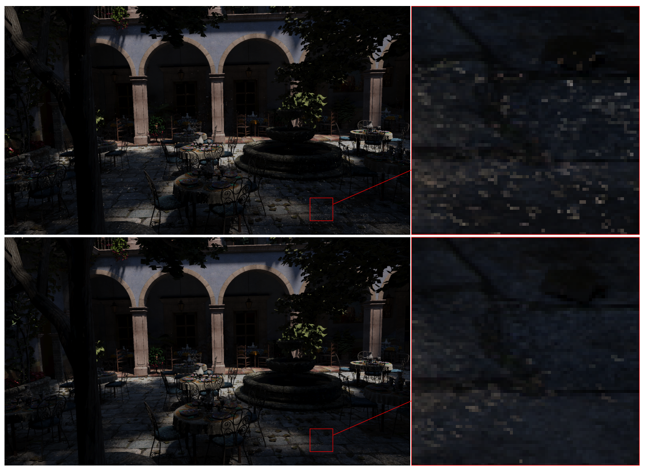

Figure 14: *Importance sampling (bottom) compared to uniform sampling (top) in the san-miguel scene at* $\alpha = 15\%$. *Importance sampling removes many of the light leaks.*

Furthermore, as we need to combine the reduced columns of Q rather than the original, the algorithm needs to be performed on Q rather than Q_Ω, which is more expensive as Q is larger.

Figure 12 shows the accuracy of the boolean matching method is compared to standard AMC and boolean Gaussian elimination across three scenes as the total number of samples increase. We found that our boolean matching method both performs similarly, and improves as the number of samples increases at roughly the same rate as both standard AMC and boolean Gaussian Elimination, showing that accuracy is not lost despite the cheaper approximation.

5.7. Importance sampling

Figure 13 shows our importance sampling method compared to uniform sampling across three scenes. We found that the algorithm performs significantly better with importance sampling, both reducing error and drastically reducing the number of darkening and light leak artefacts in scenes such as San Miguel (Figure 14).

One possible issue with our importance sampling method is that we are only splitting the matrix depending on the quadrant of the direction vector between the VPL and the slice centroid. This could lead to degenerate cases where VPLs within the same quadrant do not correlate to each other due to complex occlusions, which would result in a poor sampling distribution. This could explain why importance sampling in the kitchen scene performs poorly, as one of the light sources is hidden from most of the scene.

5.8. Adapting row sample rates

Figure 15 shows how our adaptive sampling strategy improves the accuracy compared to static sampling by removing light-leaking artefacts, whilst requiring less overall samples. This strategy also eliminates the need to tune row sample-rates for each scene as it automatically detects low-rank areas. A table detailing statistics of our comparisons is available in the supplemental.

5.9. Limitations

Completing V has the major limitation in that regions where the VPLs' visibility vary slightly over a large area (ie. penumbral re-

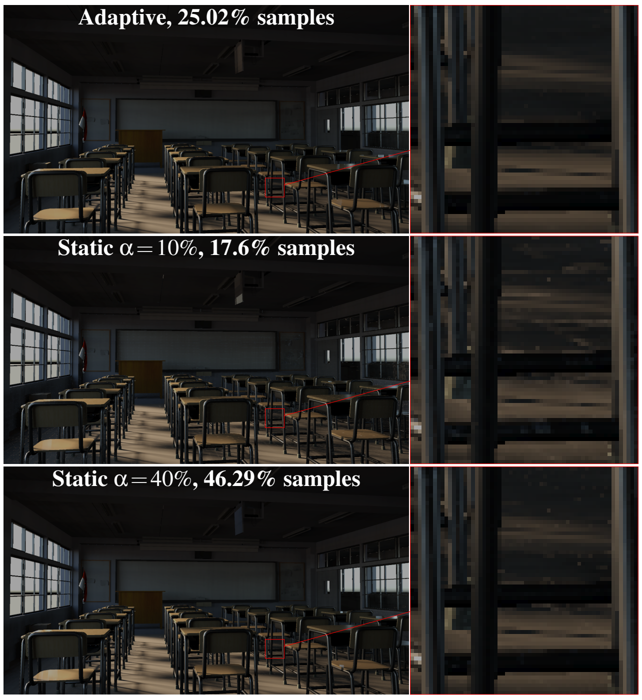

Figure 15: *Adaptive row sample-rates (top) is far more efficient than static sampling (middle, bottom) at removing light-leak artefacts in the classroom scene.*

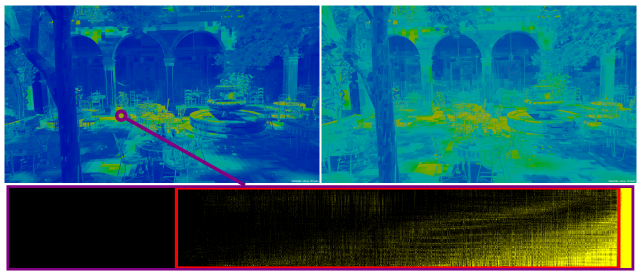

Figure 16: *Top left: rank of visibility matrix for each slice (blue = lowest rank, red = high rank). Top right: number of visibility samples computed for each slice. It is correlated with the rank of visibility matrix. Regions with higher rank ususall correspond to penumbra regions, with small variations across many columns. Bottom: sample visibility matrix from a penumbra region. Columns boxed in red all have variations.*

gions) are very high rank (Figure 16). The original Many-Lights matrix does not have this issue as smooth areas such as this are approximately low rank as the VPLs' contributions to the slice can be roughly approximated by neighbours. Low energy VPLs also do not tend to contribute much to the rank in the Many-Lights matrix, but all VPLs contribute equally to V. Choosing to complete V versus the full Many-Lights matrix is thus a trade-off, where one gains scalability and robustness over possibly lower rank matrices in diffuse scenes.

The full column sampling performed when a sub-sampled column is not represented by the current basis is also inefficient. We would ideally only sample the areas that are different. This also means that this method currently does not scale well to slice size, as increasing this also increases the rank of the matrix by possibly incorporating distant high-rank information. A more localized completion process can alleviate this issue.

Finally, the clustering method we use does not account for visibility, and can have suboptimal refinement in scenes where large numbers of lights are occluded. This is usually not visible on diffuse surfaces, but glossier surfaces make it apparent. Incorporating visibility into the clustering can resolve this.

6. Conclusions and Future Work

We have presented a method to complete the visibility within a scene using a modified version of Adaptive Matrix Completion. For a given Many-Lights scene, it only needs a fraction of the total visibility samples to accurately approximate the actual visibility between receivers and lights, resulting in roughly over a 3 times speedup compared to methods such as IlluminationCut and LightSlice, and converging better and being more general than Matrix Separation.

One interesting extension of this work is area lights. We only show the method being used with a discrete set of point lights, but the algorithm is progressive and one can simply add columns based on new light sample points. Another extension is to allow for more localized completion, and allow for early termination of row sampling if we detect no variation in a column. Finally, we aim to investigate how matrix completion can be transfered to other rendering techniques, such as path-tracing.

References

[Bit16] Bitterli, Benedikt. *Rendering resources*. https://benedikt-bitterli.me/resources/. 2016 7.

[BMB] Bus, Norbert, Mustafa, Nabil H, and Biri, Venceslas. *IlluminationCut source code*. https://perso.esiee.fr/~mustafan/Software/IlluminationCut/. Accessed: 2020-04-14 7.

[BMB15a] Bus, Norbert, Mustafa, Nabil H, and Biri, Venceslas. "Global Illumination Using Well-Separated Pair Decomposition". *Computer Graphics Forum* 34.8 (2015), 88–103 2.

[BMB15b] Bus, Norbert, Mustafa, Nabil H, and Biri, Venceslas. "IlluminationCut". *Computer Graphics Forum* 34.2 (2015), 561–573 2.

[BRA06] Ben-Artzi, Aner, Ramamoorthi, Ravi, and Agrawala, Maneesh. "Efficient shadows for sampled environment maps". *Journal of Graphics Tools* 11.1 (2006), 13–36 3.

[CCS10] Cai, Jian-Feng, Candès, Emmanuel J, and Shen, Zuowei. "A singular value thresholding algorithm for matrix completion". *SIAM Journal on optimization* 20.4 (2010), 1956–1982 3.

[CR09] Candès, Emmanuel J and Recht, Benjamin. "Exact matrix completion via convex optimization". *Foundations of Computational Mathematics* 9.6 (2009), 717 2, 3.

[DWB*06] Donikian, Michael, Walter, Bruce, Bala, Kavita, et al. "Accurate direct illumination using iterative adaptive sampling". *IEEE Transactions on Visualization and Computer Graphics* 12.3 (2006), 353–364 3.

[Faz02] FAZEL, MARYAM. "Matrix rank minimization with applications". PhD thesis. Stanford University, 2002 3.

[FBG02] FERNANDEZ, SEBASTIAN, BALA, KAVITA, and GREENBERG, DONALD P. "Local Illumination Environments for Direct Lighting Acceleration." *Rendering Techniques* 2002 (2002), 13th 3.

[GKPS12] GEORGIEV, ILIYAN, KŘIVÁNEK, JAROSLAV, POPOV, STEFAN, and SLUSALLEK, PHILIPP. "Importance caching for complex illumination". *Computer Graphics Forum* 31.2pt3 (2012), 701–710 3.

[GTZ97] GOREINOV, SERGEI A, TYRTYSHNIKOV, EUGENE E, and ZAMARASHKIN, NICKOLAI L. "A theory of pseudoskeleton approximations". *Linear algebra and its applications* 261.1-3 (1997), 1–21 2, 3.

[HDG99] HART, DAVID, DUTRÉ, PHILIP, and GREENBERG, DONALD P. "Direct illumination with lazy visibility evaluation". *Proceedings of the 26th annual conference on Computer graphics and interactive techniques*. SIGGRAPH '99. 1999, 147–154 3.

[HH09] HALDAR, JUSTIN P and HERNANDO, DIEGO. "Rank-constrained solutions to linear matrix equations using powerfactorization". *IEEE Signal Processing Letters* 16.7 (2009), 584–587 2, 3.

[HKWB09] HAŠAN, MILOŠ, KŘIVÁNEK, JAROSLAV, WALTER, BRUCE, and BALA, KAVITA. "Virtual spherical lights for many-light rendering of glossy scenes". *ACM Transactions on Graphics (TOG)* 28.5 (2009), 1–6 6.

[HPB07] HAŠAN, MILOŠ, PELLACINI, FABIO, and BALA, KAVITA. "Matrix Row-column Sampling for the Many-light Problem". 26.3 (July 2007), 26fffdfffdfffdes. DOI: 10.1145/1276377.1276410 1, 2, 4.

[HR10] HUANG, FU-CHUNG and RAMAMOORTHI, RAVI. "Sparsely precomputing the light transport matrix for real-time rendering". *Computer Graphics Forum* 29.4 (2010), 1335–1345 2.

[HWH*16] HUO, YUCHI, WANG, RUI, HU, TIANLEI, et al. "Adaptive matrix column sampling and completion for rendering participating media". *ACM Transactions on Graphics (TOG)* 35.6 (2016), 167 2.

[HWJ*15] HUO, YUCHI, WANG, RUI, JIN, SHIHAO, et al. "A matrix sampling-and-recovery approach for many-lights rendering". *ACM Transactions on Graphics (TOG)* 34.6 (2015), 210 2.

[Jak10] JAKOB, WENZEL. *Mitsuba renderer*. http://www.mitsuba-renderer.org. 2010 7.

[Jen01] JENSEN, HENRIK WANN. *Realistic image synthesis using photon mapping*. AK Peters/CRC Press, 2001 4.

[Kel97] KELLER, ALEXANDER. "Instant Radiosity". *Proceedings of the 24th Annual Conference on Computer Graphics and Interactive Techniques*. SIGGRAPH '97. 1997, 49–56. DOI: 10.1145/258734.258769 2.

[KS14] KRISHNAMURTHY, AKSHAY and SINGH, AARTI. "On the power of adaptivity in matrix completion and approximation". *arXiv preprint arXiv:1407.3619* (2014) 2, 3.

[McG17] MCGUIRE, MORGAN. *Computer Graphics Archive*. https://casual-effects.com/data. July 2017. URL: https://casual-effects.com/data 7.

[MMB*18] MARIA, MAXIME, MUSTAFA, NABIL, BARDOUX, THOMAS, et al. "Visibility based WSPD for Global Illumination". *13th International Joint Conference on Computer Vision, Imaging and Computer Graphics Theory and Applications (VISIGRAPP 2018)*. Vol. 1. 2018, 81–90 2.

[OP] OU, JIAWEI and PELLACINI, FABIO. *Lightslice source code*. https://jiawei.ooo/lightslice. Accessed: 2020-03-23 7.

[OP11] OU, JIAWEI and PELLACINI, FABIO. "LightSlice: matrix slice sampling for the many-lights problem". *ACM Transactions on graphics (TOG)* 30.6 (2011), 179 1, 2.

[RD16] REHFELD, HAUKE and DACHSBACHER, CARSTEN. "Lightcut interpolation". *Proceedings of High Performance Graphics*. 2016, 99–108 2.

[RGKM07] RITSCHEL, TOBIAS, GROSCH, THORSTEN, KAUTZ, JAN, and MÜELLER, STEFAN. "Interactive illumination with coherent shadow maps". *Proceedings of the 18th Eurographics conference on Rendering Techniques*. Eurographics Association. 2007, 61–72 3.

[RGKS08] RITSCHEL, TOBIAS, GROSCH, THORSTEN, KAUTZ, JAN, and SEIDEL, HANS-PETER. "Interactive global illumination based on coherent surface shadow maps". *Proceedings of Graphics Interface 2008*. Canadian Information Processing Society. 2008, 185–192 3.

[TW16] TANNER, JARED and WEI, KE. "Low rank matrix completion by alternating steepest descent methods". *Applied and Computational Harmonic Analysis* 40.2 (2016), 417–429 3.

[VGS*19] VIBERT, NICOLAS, GRUSON, ADRIEN, STOKHOLM, HEINE, et al. "Scalable Virtual Ray Lights Rendering for Participating Media". *Computer Graphics Forum* 38.4 (2019), 57–65 2.

[VKK18] VÉVODA, PETR, KONDAPANENI, IVO, and KŘIVÁNEK, JAROSLAV. "Bayesian online regression for adaptive direct illumination sampling". *ACM Transactions on Graphics (TOG)* 37.4 (2018), 125 3.

[WABG06] WALTER, BRUCE, ARBREE, ADAM, BALA, KAVITA, and GREENBERG, DONALD P. "Multidimensional lightcuts". *ACM Transactions on graphics (TOG)* 25.3 (2006), 1081–1088 2.

[WC13] WU, YU-TING and CHUANG, YUNG-YU. "VisibilityCluster: Average directional visibility for many-light rendering". *IEEE Transactions on Visualization and Computer Graphics* 19.9 (2013), 1566–1578 3.

[WDT*09] WANG, JIAPING, DONG, YUE, TONG, XIN, et al. "Kernel Nyström method for light transport". *ACM Transactions on Graphics (TOG)* 28.3 (2009), 29 2.

[WFA*05] WALTER, BRUCE, FERNANDEZ, SEBASTIAN, ARBREE, ADAM, et al. "Lightcuts: a scalable approach to illumination". *ACM Transactions on graphics (TOG)* 24.3 (2005), 1098–1107 1, 2.

[WKB12] WALTER, BRUCE, KHUNGURN, PRAMOOK, and BALA, KAVITA. "Bidirectional lightcuts". *ACM Transactions on Graphics (TOG)* 31.4 (2012), 59 2.

[WYZ12] WEN, ZAIWEN, YIN, WOTAO, and ZHANG, YIN. "Solving a low-rank factorization model for matrix completion by a nonlinear successive over-relaxation algorithm". *Mathematical Programming Computation* 4.4 (2012), 333–361 3.