Improving Cache Placement for Efficient Cache-based Rendering

Yu-Ting Wu[1*] and I-Chao Shen[2]

[1*]Computer Science Department, National Taipei University .
[2]Computer Science Department, The University of Tokyo .

*Corresponding author(s). E-mail(s): yutingwu@mail.ntpu.edu.tw;
Contributing authors: ichaoshen@g.ecc.u-tokyo.ac.jp;

Abstract

This paper proposes a new method to improve cache placement for various rendering algorithms using caching techniques. The proposed method comprises two stages. The first stage computes an initial cache distribution based on shading points' geometric proximity. We present a view-guided method to cluster shading points based on their world-space positions and surface normals, while considering the camera view to avoid producing small clusters in the final image. The proposed method is more robust and easier to control than previous shading point clustering methods. After computing the shading functions at the initial cache locations, the second stage of our method utilizes the results to allocate additional caches to regions with shading discontinuities. To achieve this, a discontinuity map is created to identify these regions and used to insert new caches based on importance sampling. We integrate the proposed method into several cache-based algorithms, including irradiance caching, importance caching, and ambient occlusion. Extensive experiments show that our method outperforms other cache distributions, producing better results both numerically and visually.

Keywords: ray tracing, importance caching, irradiance caching, ambient occlusion.

1 Introduction

Rendering is a time-consuming process as it requires evaluating the high-dimensional integral of the shading function at numerous image pixels. To minimize shading expenses, a commonly employed technique is to exploit the spatial coherence of the rendered images through ***caching***: evaluating the costly function at sparse locations known as caches, and reusing the results through interpolation or extrapolation at other surface points. Caching techniques can expedite computations of both direct and indirect illumination by storing various data such as (ir)radiance [1–13], visibility [14], or the importance sampling function [15, 16].

The distribution of caches has a significant impact on the final rendering quality. To accurately predict the target function, the density of caches should be proportional to the complexity of the rendered image. Regions of high complexity, such as small geometry features or shadow boundaries, require more caches to capture the rapidly changing signal. However, obtaining the optimal distribution before the actual rendering process is challenging. Therefore, some methods [1–3, 6, 7, 9, 12, 14] start with random or uniform caches across the image and generate new caches if reliable ones are not found in a shading point's local neighborhood. This update-on-demand strategy; however, has two major challenges. Firstly, the update criteria are application-dependent and not all applications possess an accurate error estimation for cache interpolation. Secondly, because all the caches need to be stored in an acceleration structure such as Octree or Kd-Tree for efficient queries, any rendering-time update to the structure requires a read-write lock, which greatly reduces parallelism [4].

An alternative strategy is to improve the cache placement prior to rendering. Most methods in this category focus on specific applications [4, 5, 8, 11, 16]. A more general method in this context is spatial-directional clustering (**SD**) [17], which was first proposed by Ou and Pellacini to cluster shading points for virtual point light (VPL) rendering and has later been leveraged by various applications [16, 18–20]. The method first builds a 6D bounding volume hierarchy (BVH) to group shading points where the 6D coordinates encompass the 3D coordinate and surface normal at each shading point. The BVH is continually split until the maximum extent of the 6D bound is below a predefined threshold or the number of shading points within the bound is below a predefined threshold. Lastly, one cache is generated by sampling a shading point within a bound (shading point cluster). However, this method has two major drawbacks. Firstly, controlling the threshold for cluster splitting with a single value for both the spatial and angular properties is challenging. For example, a threshold value of one might be very small for the spatial difference in a large scene, but too large for the difference of surface normals. Secondly, selecting the threshold of cluster size for preventing small clusters faces a dilemma. As demonstrated in Figure 1, using various configurations, the method suffers from artifacts in distant and near regions, respectively.

This paper proposes a new method for determining a good cache distribution in the early stage (*i.e.*, prior to rendering) for various cache-based rendering techniques. The proposed algorithm comprises two stages. In the first stage, we introduce a new shading point clustering method called view-guided clustering (**VGC**) to generate a set of initial cache locations based on the geometric proximity of the shading points. Our method allows for more intuitive parameter control and can robustly adapt to camera view, striking a better balance in 3D geometric proximity and image-space importance. As shown in Figure 1, our **VGC** produces better visual quality and lower numerical errors than existing methods.

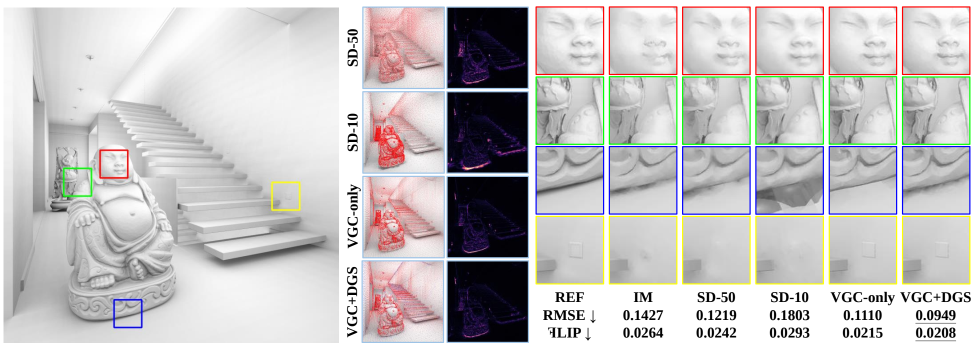


Fig. 1: Equal-cache comparison (30K caches) on STAIRCASE scene rendered with cached-based ambient occlusion [14] using various cache distributions: image-space uniform sampling (**IM**), spatial-directional clustering [17] with a minimum cluster size of 50 (**SD-50**) and 10 (**SD-10**), our view-guided clustering only (**VGC-only**), and our full method (**VGC+DGS**). The middle columns visualize each method's corresponding cache distribution and a perceptual error computed by ꟻLIP [21]. As shown from the right insets, image-space uniform sampling (**IM**) fails to capture geometric properties. Spatial-directional clustering (**SD**) encounters troubles in parameter selection. Using a large cluster size blurs out the details of the distant Buddha, while using a small cluster size results in large errors in near regions. Our view-guided clustering (**VGC**) strikes a better balance in 3D geometric proximity and image-space importance, achieving better visual quality and smaller root-mean-squared error (RMSE) and perceptual error. Our full method (**VGC+DGS**) with a discontinuity-guided sampling as the second pass further enhances the results, especially in regions with shading change.

While the proposed view-guided clustering improves cache distributions by considering geometric proximity, it does not account for changes due to shading, as shown in the blue inset of Figure 1. To address these shading discontinuities, we devise a discontinuity-guided sampling (**DGS**) algorithm as the second stage building on the results obtained in the previous stage. First, we compute a discontinuity map to identify regions with potentially high shading and geometry discontinuities. Based on the discontinuity map, we then use importance sampling to deposit adaptive caches in challenging regions, such as shading edges. We redistribute a fraction of cache budget from **VGC** to **DGS**. The final rendered results using our full method (**VGC+DGS**) exhibit improvement in shadow regions and surface details.

The proposed method provides a better cache distribution with a small computation overhead. Through comprehensive experiments, we demonstrate that our method enhances the efficiency of cache-based applications, including irradiance caching [1], importance caching [15], and ambient occlusion rendered with visibility caches [14]. The improved cache distribution can also benefit methods using an update-on-demand strategy by offering a better initial cache distribution.

2 Related Work

This section briefly reviews the existing cache-based rendering techniques. For each cached-based algorithm, we first introduce its target and then discuss its cache-placement strategy. We will focus on methods that exploit the coherence of the rendering functions between shading points. Other methods that cache light path contributions such as VPL [22] or photon mapping [23] are less relevant to our work and not included here.

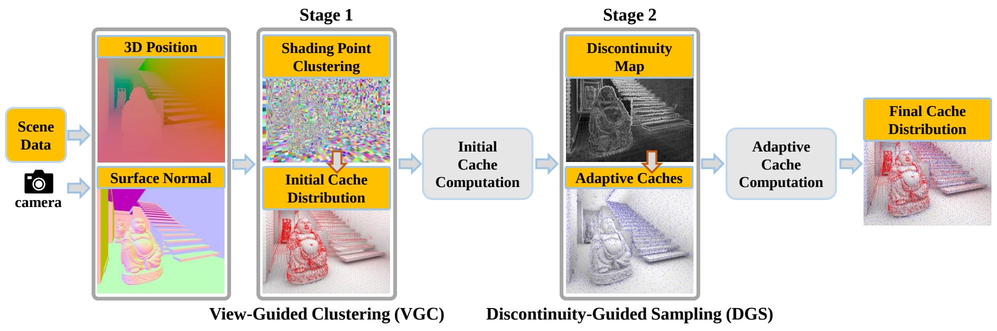
Irradiance caching and radiance caching. As a pioneer of the caching techniques, Ward *et al.* [1] proposed irradiance caching to accelerate the computation of indirect illumination. Their method exploits the low-frequency characteristic of indirect lighting by computing the irradiance values at a set of sparse locations sampled uniformly across the image. During rendering, new caches are inserted on-demand if the potential error of interpolating existing caches is larger than a threshold. The interpolation criterion is later improved by the follow-up works, including the use of translational gradients of irradiance [2] and second-order continuity based on Hessian [6]. Tabellion and Lamorlette [3] accelerate the computation of irradiance caches with an approximate lighting model. Brouillat *et al.* [5] construct caches by utilizing photon maps. Gautron *et al.* [4] present a GPU-friendly reformulation.

The radiance caching proposed by Křivánek *et al.* [7] extended irradiance caching to handle glossy materials by storing directional radiance functions instead of irradiance. Then several approaches have been presented to improve the strategies for interpolation and extrapolation [8, 9, 12]. Radiance caching has also been reformulated using a probe representation [24] or pre-convolution [10, 11], or extended for rendering participating media [25, 26] and animation [27]. Recently, learning-based methods have also been explored to improve radiance cache for real-time rendering [13].

The Lumen in Unreal Engine 5 (UE5) [28] combines screen-space radiance caches [7] and world-space radiance caches [29] to render global illumination. The two types of caches are combined according to the distance of light sources. To efficiently generate caches in each frame, the screen-space caches are placed regularly in image grids with an adaptive division to capture discontinuities. The world-space radiance caches are placed uniformly in world space. Compared to our method, they combine several techniques and heavily rely on temporal filtering to achieve real-time performance, while our work concentrates on improving the cache distribution for offline rendering.

Recently path tracing has become the most popular approach due to its simplicity, generality, and unbiased characteristics. However, it often produces results with noticeable noise and requires numerous samples to converge. ReSTIR [30] and its following works [31] have greatly improved the sampling efficiency by reusing spatial-temporal samples with the reservoir resampling algorithm. Because the ReSTIR algorithms focus on improving the sampling function at a surface location, it can be combined with cache-based rendering to compute cache values at shading points [13].

Visibility caching. Clarberg and Akenine-Moeller [14] compute and store the visibility functions at sparse locations called visibility caches and use them to interpolate the visibility function at every shading point. They present two rendering applications based on visibility caches. First, they reformulate the rendering equation with



Stage 1
Stage 2
Scene Data
camera
3D Position
Surface Normal
Shading Point Clustering
Initial Cache Distribution
Initial Cache Computation
Discontinuity Map
Adaptive Caches
Adaptive Cache Computation
Final Cache Distribution
View-Guided Clustering (VGC)
Discontinuity-Guided Sampling (DGS)

Fig. 2: **Overview of the proposed method.** Our method comprises two stages. In the first stage, we present a new view-guided method (**VGC**) for clustering shading points. In each shading cluster, a shading point is selected as the cache location, determining an initial cache distribution. To further identify regions with complex shading that are not captured in the first stage, the second stage (**DGS**) first utilizes the rendering results at initial caches and generates a discontinuity map. Then additional caches are generated by importance sampling the map. The initial and adaptive caches constitute the final cache distribution for cache-based rendering. Here, we demonstrate an example of ambient occlusion using visibility caches [14]. We can integrate our two-stage sampling method into other cached-based techniques such as irradiance caching [1] or importance caching [15].

the interpolated visibility function integrated into the sampling function using control variate. Secondly, they demonstrate that the visibility caches can be used to produce noise-free ambient occlusion.

Importance caching. Georgiev *et al.* [15] proposed importance caching to handle complex lighting and visibility in the context of VPL-based rendering. Prior to rendering, they first compute the contributions from all VPLs to a set of randomly sampled locations across the image. The cached VPL contributions are then built into several sampling distributions and combined using multiple importance sampling at each shading point during rendering. Yoshida *et al.* [16] later improved the idea with a better cache distribution.

To categorize the cache placement strategy, most irradiance caching [1–3, 6], radiance caching [7, 9, 12], and visibility caching [14] methods start with a uniform distribution and insert new caches in the course of rendering based on error estimation. For other methods that generate cache distribution prior to rendering, importance cachine [15] places caches randomly in image space. Scherzer *et al.* [10] adopt a fixed cache distribution based on blue noise sampling in world space. Brouillat *et al.* [5] generate irradiance caches on photon locations according to a photon map. Gautron *et al.* [4] and Křivánek *et al.* [8] refine uniform sampling by adding additional caches based on error estimation at radiance caches. Yoshida *et al.* [16] proposed a two-pass approach to improve the distribution of importance caches. We discuss the major differences between our method and their approaches in the Section 3.

3 Algorithm

Figure 2 demonstrates the flowchart of our method. To exploit the spatial coherence of the shading function in the rendered images, we first obtain the set of shading points by determining the 3D intersections of the scene and the camera rays. For each shading point, we store its 3D position and surface normal. Due to no shading information being available at the beginning, our first stage algorithm, view-guided clustering (**VGC**), utilizes the geometry proximity to compute an initial cache distribution. After computing the shading function at these cache locations, the second stage, discontinuity-guided sampling (**DGS**), generates additional caches by utilizing the shading information from the initial caches. The final cache distribution is then used for cache-based rendering.

3.1 First Stage: View-Guided Clustering

The goal of the first stage is to generate a set of uniform caches based on the geometric proximity of shading points. To consider both spatial and angular properties, we first map the geometry data of a shading point onto a 5D coordinate, encompassing the 3D world-space position (x, y, z) and 2D spherical angles ϕ and θ for the surface normal. We then build a bounding volume hierarchy (BVH) for all shading points in a top-down fashion. Starting from the root, we iteratively split the BVH node into two children if the difference in geometric properties within the node is larger than predefined thresholds. Unlike **SD** [17] that uses a single threshold to control spatial and angular splitting, we adopt separate thresholds for more intuitive control: the spatial threshold σ_s constrains the 3D extent of a node's bounding box by meters, while the angular threshold σ_a constrains the maximum difference of surface normals in a node by angles.

Given the 5D bounding volume of all shading points, our method first splits the BVH according to the spatial property (*i.e.*, 3D world-space position). We replace a node with its two children if its longest world-space axis (either in x, y, or z direction) is larger than the spatial threshold σ_s. Next, for all leaf nodes generated by spatial splitting, we recursively replace them with their two children if the difference of surface normals in a node is larger than the angular threshold σ_a. Once the angular splitting process is complete, all the leaf nodes form the final shading point clusters.

For cache-based rendering, most of the computation time is spent on evaluating cache data. Consequently, making each cache effect as many shading points as possible improves rendering efficiency. In clustering-based methods such as **SD** [17] and our **VGC**, this can be implemented by avoiding generating small clusters that are projected to a few pixels in the final rendered image. Unlike **SD** that simply limits the cluster size with a pre-defined threshold, we propose a more robust way by adjusting the two thresholds at each node's splitting:

$$\begin{aligned} \sigma_s' &= \sigma_s + (1-\alpha)D^2\sigma_s, \\ \sigma_a' &= \sigma_a + (1-\alpha)D^2\sigma_a, \end{aligned} \tag{1}$$

where $D = (d - d_{\min})/(d_{\max} - d_{\min})$ with d, $d_{\min}$, and $d_{\max}$ being the depth of the node and the minimum and maximum depth of all shading points, respectively. α is a parameter that controls the maximum scaling of the thresholds. We set α to $(1+\Gamma(d_{\max} - d_{\min}))$ with the parameter λ controlling the decaying of importance with respect to depth. We use $\Gamma = 0.2$ for all scenes in this paper. Equation 1 uses the original thresholds σ_s and σ_a for the shading points with depth $d_{\min}$ and $\alpha \cdot \sigma_s$ and $\alpha \cdot \sigma_a$ for the ones with depth $d_{\max}$. For the shading points with depth in between, the thresholds are scaled proportional to the squared distance to the camera because the projected area is inversely proportional to the square of the distance.

Once the clustering is done, a cache is generated by selecting a member in the cluster. We first compute the average 3D coordinate p_{avg} and surface normal n_{avg} of all members in the cluster, with a small jittering to avoid structural artifacts. Then the closest shading point to the average geometric properties is chosen as the cache location based on the distance metric:

$$\text{Dist}(s, avg) = \|p_s - p_{avg}\| + \lambda\sqrt{1 - n_s \cdot n_{avg}}, \tag{2}$$

where the p_s and n_s denote the 3D coordinate and surface normal of the shading point, λ denotes a parameter that trades off the importance of Euclidean and angular distance. We set λ to $0.5/L$ where L is the diagonal of the scene's bounding box [15].

In Figure 1, we compare the cache distributions generated by our method and spatial-directional BVH (**SD**) [17]. We also show the image-space uniform sampling (**IM**) for comparison. The images are ambient occlusion results rendered with the corresponding cache distribution and the **VisibilityCache** algorithm (presented in Section 4.3). **IM** disregards geometric properties and as a result, it produces significant errors on curved surfaces, such as the two Buddhas. For **SD**, we tested it using various constraints on the minimum size of a shading cluster. The results indicated that a large cluster size (50 for **SD-50**) failed to capture the high-frequency details of the distant Buddha. Conversely, a smaller cluster size (10 for **SD-10**) placed too many caches on distant surfaces, leading to insufficient cache density in nearby regions. The red and green insets of **SD-50** and the blue and yellow insets of **SD-10** in Figure 1 demonstrate such scenarios, respectively. Our view-guided clustering, **VGC**, strikes a better balance between the 3D geometric proximity and image-space importance.

3.2 Second Stage: Discontinuity-Guided Sampling

The distribution in the first stage only considers geometric properties. Thus, it cannot capture discontinuities due to shading. One example is the shadows below the front Buddha disappeared as shown in Figure 1. For this reason, we devise a second stage to detect regions with complex shading, and redistribute a fraction of samples to these regions.

To collect shading information, we first compute the target shading functions (ambient occlusion, irradiance value, importance function, etc.) at the initial cache locations based on various caching-based applications. For each cache, in addition to storing the function value, we also store the individual application-dependent gathering sample to provide more information on local shading behavior:

Irradiance caching. The irradiance value at a cache is computed by sampling the radiance from a set of directions using the equal-area spherical mapping [32]. At each cache, we store the accumulated irradiance value as well as the individual radiance value along each sampling direction.
Importance caching. We store the accumulated and individual contributions from all virtual lights at each cache as in the original method [15].
Ambient occlusion. The ambient occlusion value is computed by sampling the visibility function with a set of shadow rays using the equal-area spherical mapping [32]. At each cache, we store the ambient occlusion value as well as the individual visibility result in a binary bitmask.
It is worth noting that this process only introduces minimal memory usage without additional computational cost.

After the shading function is computed at the sparse caches, we create a per-pixel discontinuity map based on the gathering samples to identify regions with large shading discontinuities. For the shading point corresponding to each pixel, we search its K nearest caches and compute the average difference Diff_{avg} and the maximal difference Diff_{max} between these K caches. The difference between two caches is computed by summing the absolute difference of their corresponding gathering samples. Diff_{avg} and Diff_{max} tend to be higher in regions with complex shading, such as shadow boundaries. In addition to shading discontinuities, we observed that the first stage may struggle to identify the discontinuities of the shading function at regions with small geometry features, especially when the local density of caches is insufficient. For example, **VGC** fails to render the ambient occlusion on the cabinets correctly in Bathroom shown in Figure 6. To address this problem, we include a geometry term G to detect small geometry discontinuities at a shading point s:

$$G(s) = \sum_{q \in \Omega} (1 - n_s \cdot n_q), \tag{3}$$

where q is a nearby shading point within a local window of s. n is the surface normal of the shading points. We use a 5×5 block for the local window. Putting all terms together, the per-pixel discontinuity measurement $D(s)$ of a shading point s is then defined as:

$$D(s) = \text{Diff}_{avg}(s) + \text{Diff}_{max}(s) + G(s). \tag{4}$$

$\text{Diff}_{avg}(s)$, $\text{Diff}_{max}(s)$, and $G(s)$ are normalized to the range $[0, 1]$. To prevent creating caches too close to an existing one, we set the $D(s)$ of a shading point to zero if it has already been chosen as a cache.

To efficiently generate a set of caches based on the discontinuity map, we first importance sample a row of the map using its marginal density. After choosing a row, we importance sample a pixel within the row using conditional probability. The selected pixel becomes a new cache. Figure 3 demonstrates the discontinuity map, adaptive cache locations, and visual comparisons of using **VGC**-only and **VGC+DGS** on the Statues scene (left) and the Robots scene (right). In the left part of the image, the shadow regions on the objects' surface and ground produce larger discontinuity values in the discontinuity map, thus receiving more adaptive caches. The insets showcase that the artifacts in shadow regions are reduced thanks

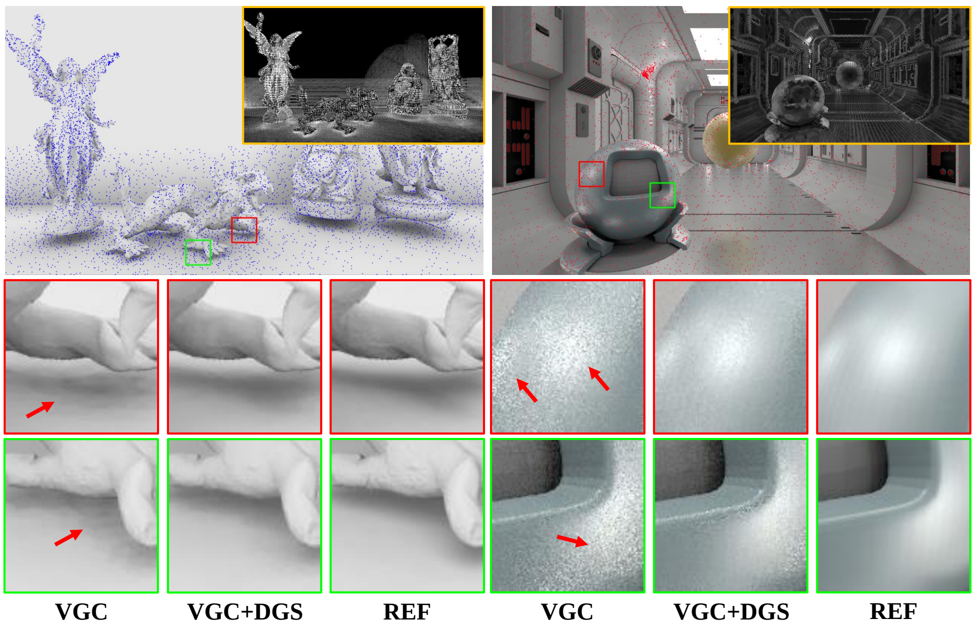


Fig. 3: **The effectiveness of Discontinuity-Guided Sampling (DGS)**. The left part of the image demonstrates the STATUES scene rendered using ambient occlusion with 50K caches; the right part demonstrates the ROBOTS scene rendered using importance caching with 25K caches. In both scenes, we demonstrate the discontinuity maps (the gray-scale insets), the adaptive caches generated by the **DGS** stage (blue and red dots on the images), and visual comparisons of **VGC**-only and **VGC+DGS**. For **VGC+DGS**, we shift 15% caches (7.5K for STATUES and 3.75K for ROBOTS) from the **VGC** stage to the **DGS** stage, making the total number of caches roughly the same as in **VGC**-only. The two discontinuity maps demonstrate that the regions with larger shading discontinuities (*e.g.* shadows or glossy surfaces) have larger values, thus receiving more adaptive caches. Insets showcase improvements in visual quality thanks to the **DGS** stage.

to the increasing cache density. In the right part of the image, **VGC** produces large errors on the robots' surface because its geometry-based clustering fails to capture the discontinuities of glossy shading. The discontinuity map captures the large shading discontinuities and places more adaptive caches on the robots' surfaces. The insets showcase noise reduction by incorporating the **DGS** stage.

There are a few methods that also adopt a two-pass strategy to improve the cache distribution [8, 16]. They first compute a set of initial caches based on uniform sampling or shading point clustering [17]. Then they loop over the initial caches and examine how they differ from the neighboring caches. New caches are inserted if the estimated error is larger than a threshold. Because the refinements are performed per cache, we observed that their approaches might fail to capture the shading discontinuities when the initial caches are insufficient. Moreover, selecting the thresholds for inserting new caches is not intuitive because they are application-dependent and scene-dependent. For example, in our experiment the threshold for **SD+AC** [16] to generate 2.5K adaptive caches for STATUES, ROBOTS, and the CRYTEKSPONZA scene in Figure 4 are 6000, 93000, and 60, respectively. Compared to these methods, our view-guide clustering is more effective than uniform sampling and the spatial-directional clustering they used in the first pass. Our importance sampling approach offers a united manner to generate new caches at pixels with relatively large potential errors.

3.3 Run-time Rendering

During rendering, we employ a strategy similar to the one used in VisibilityCache [14]. In order to find a small number of caches M for estimating the target function for a shading point s, we first perform a range search in the acceleration structure and locate N spatially nearest caches in the world space. After filtering out the ones with dissimilar surface normals with a threshold θ_{max}, we compute the distance $\text{Dist}(c^i, s)$ between a remaining cache c^i and the shading point using:

$$\text{Dist}(c^i, s) = \left\|p_c^i - p_s\right\| + \lambda\sqrt{1 - n_c^i \cdot n_s}, \tag{5}$$

where p_c^i, p_s, n_c^i, and n_s are the 3D coordinates and surface normals of the cache c^i and the shading point s, respectively. We decide the value of λ according to Equation 2. We only use the M caches with the smallest distance for the subsequent computation. For applications that directly interpolate cache values such as irradiance caching (Section 4.1) and ambient occlusion (Section 4.3), the value at the shading point is computed by weighted averaging the M cache values, with the inverse of the distance defined in Equation 5 as the weight. For importance caching (Section 4.2), we apply the multiple importance sampling presented in the original paper [15] using the M caches. We set N, M, and θ_{max} to 20, 6, and 30° for all applications and results in this paper. If no caches remain after filtering, we will loosen the constraints and use all N caches as candidates for selecting M among them.

4 Experiments

We integrate our method into three cache-based techniques: irradiance caching [1], importance caching [15], and ambient occlusion using visibility caching [14], with implementations on top of the PBRT3 system [33]. We conduct equal-cache comparisons on various cache distributions, including image-space uniform sampling (**IM**), spatial-directional BVH (**SD**) [17], our view-guided clustering (**VGC**) only, and our full two-stage method (**VGC+DGS**). We also compare to the adaptive sampling approach proposed by Yoshida *et al.* [16] (**SD+AS**) although the method was initially designed for importance caching. All images in this paper are rendered in 1920×1080 resolutions with 4 sub-pixel samples for antialiasing, generated on a desktop with an Intel i7-10700 CPU at 2.90 GHz, 72GB of RAM, and using 16 threads. Reference images are rendered by computing the shading function at all pixels. We evaluate

Parameters	Description	Value
Set by users		
N_c	Total number of caches	
σ_a (Eq.1)	Threshold for the maximum angular difference in shading point clustering	Suggested to set based on N_c
Automatically derived		
σ_s (Eq.1)	Threshold for the maximum spatial difference in shading point clustering	Determined by binary searching according to σ_a and N_c
λ (Eq.2)	Control the weight of position and normal differences while selecting cache location	$0.5/L$ where L is the diagonal of the scene's bounding box
Fixed default values		
Γ	Compute α in Eq.1	0.2
N	Number of spatial nearest neighbors	20
M	Number of spatial nearest neighbors after considering the angular difference	6
$\theta_{\max}$	The maximum difference of surface normals for filtering out caches with dissimilar orientation	30°
R	Ratio of caches that is redistributed from **VGC** to **DGS**	15%

Table 1: Parameters used in the paper.

all methods using root-mean-square error (RMSE) and a perceptual-based metric, ꟻLIP [21].

Parameters. Table 1 lists the parameters and their values used in our method. Among them, N, M, and $\theta_{\max}$ are used by all cache-based methods. Default values work well for these parameters and for Γ and R of our method. σ_s and λ can be derived automatically. The only two parameters to determine are the total number of caches N_c and the angular threshold σ_a. For the angular threshold σ_a, we suggest setting it according to the total target number of caches N_c. We use 15° for 120K caches and 25° for 25K and 40K caches, respectively. Given the σ_a, our method then determines the spatial threshold σ_s to meet the total number of caches using a binary search process. Unless N_c or σ_a changes, the binary search process only has to be performed once.

For **SD**, we tune the threshold value for BVH splitting to meet the desired number of caches N_c. We also tested various constraints for the minimum cluster size and selected the one that resulted in the lowest error. If the RMSE and ꟻLIP values conflict, we choose the size that yields better visual quality.

For **SD+AS** and our **VGC+DGS**, we redistribute a ratio R of the cache budgets N_c from the first stages (**SD** and **VGC**, respectively) to the second stage. We use $R = 15\%$ for all scenes and applications in this paper.

4.1 Irradiance Caching

We compute the irradiance values at the locations of caches using $4,096$ sampling rays generated by equal-area spherical mapping [32] prior to rendering. During rendering, we first estimate the irradiance value at a shading point with the procedure introduced in Section 3.3, then the irradiance value is multiplied by a directional reflectance term for supporting moderate glossy materials.

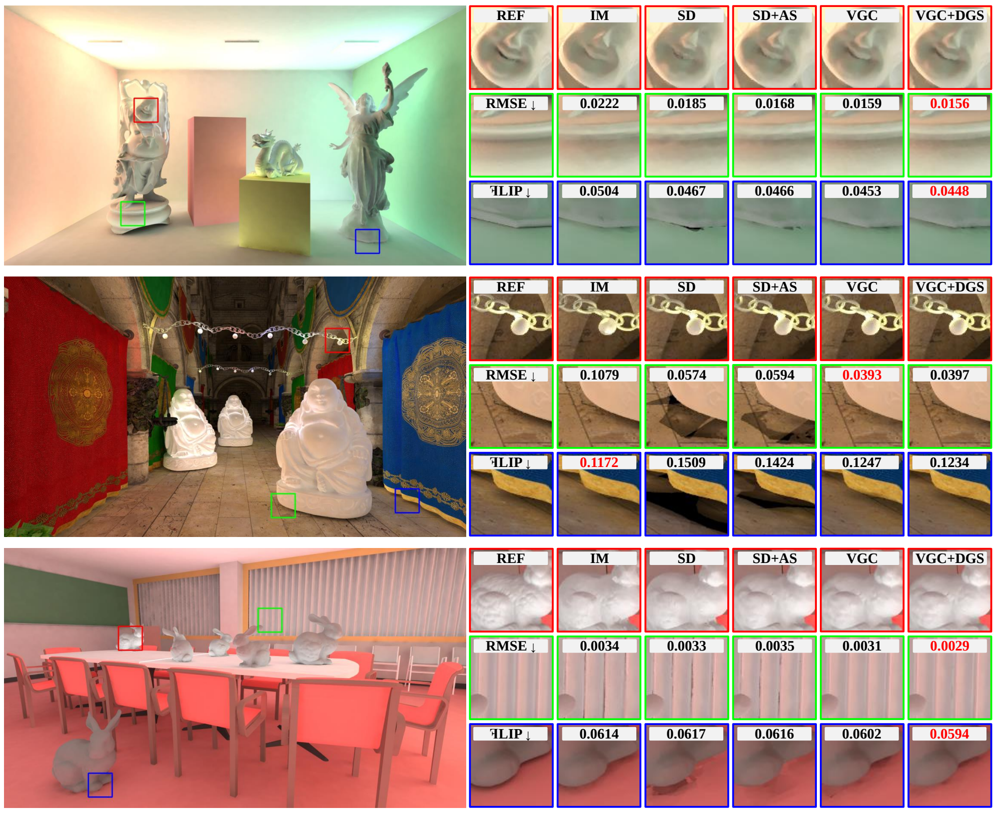


Fig. 4: Equal-cache comparison (40K caches) of irradiance caching on CornellBox, CrytekSponza, and Conference with different cache distributions: image-space caching (**IM**), spatial-directional BVH (**SD**) [17], spatial-directional BVH with adaptive sampling (**SD+AS**) [16], and our method (**VGC** only and **VGC+DGS**). The full images on the left are rendered with our full method (**VGC+DGS**). For all cases, **VGC** and **VGC+DGS** outperform **SD** and **SD+AS** both visually and numerically. Although **IM** produces the lowest perceptual error in CrytekSponza, it blurs out the specular highlight of the iron chain and the surface details on the buddhas.

Figure 4 shows the equal-sample comparisons (40K caches) of the indirect lighting results rendered with various cache distributions. For all scenes, **IM** blurs out the geometry details on objects with complex shapes. It also produces large errors on objects with fine geometry such as iron chains in the CrytekSponza, and in distant objects like the curtains in Conference. By considering the geometric properties of shading points, **SD**, **SD+AS**, and our methods (**VGC** and **VGC+DGS**) improve the shading on curved surfaces. In CornellBox, the differences between the results produced by **SD**, **SD+AS**, **VGC**, and **VGC+DGS** are subtle due to a simple layout. **SD+AC**,**VGC**, and **VGC+DGS** produce better results on the face of the Buddha model (red inset). The scene layout in CrytekSponza shows a difficult case for **SD** and **SD+AS**. They place too many samples on the curved surface, resulting in apparent artifacts for the shadows on the ground due to insufficient caches. Although **IM** produces the lowest perceptual error, it produces incorrect highlights on the iron chain and produced the highest RMSE. In Conference, **SD** and **SD+AS** smooth out the details of the distant bunny and produce artifacts on the curtains by using a larger cluster size to reduce the global error. On the other hand, **VGC** and **VGC+DGS** obtain lower errors while preserving the rendering qualities of the distant bunnies and curtains. This demonstrates that our method strikes a better balance for allocating samples under various scene conditions. Our full method does not produce apparent improvements over **VGC** alone in irradiance caching. The reason is that the indirect illumination is relatively smooth, thereby the re-allocation of samples does not produce significant impacts.

4.2 Importance Caching

Importance caching [15] computes global illumination based on the virtual point light formulation [22]. We convert direct and indirect illumination into approximately 8K virtual point lights. At each cache location, we compute the contributions from all virtual lights and build four importance sampling distributions, ranging from aggressive to conservative. During rendering, the sampling distributions stored at the nearest M caches are combined with a bilateral multiple-importance sampling scheme at each shading point to generate 4 light samples. Based on the finding of the ablation study discussed in Section 4.4, we exclude the geometry term in Equation 4 in importance caching. Figure 5 demonstrates four example scenes rendered by importance caching using 25K caches, with various cache distributions. Disregarding the geometric properties, **IM** produces the most noise on fine geometry among all methods. Similar to the case of CrytekSponza in the application of irradiance caching, **SD** and **SD+AS** face difficulties when used in Sibenik because they fail to strike a balance between spatial and angular properties. As a result, they produce noticeable artifacts on the stairs due to poor cache density. Thanks to the better cache location, our method offers more accurate sampling functions and reduces the noises and artifacts globally. In Robots scene, **IM** produces more noises in distant regions due to a lower cache density. **SD** and **SD+AC** suffer from larger noises in shadows. **VGC** produces a larger error on the robot's glossy surfaces due to the complex shading on simple geometry. The **DGS** stage in our full method compensates for the insufficient cache density on the robot's surface and produces better results.

4.3 Ambient Occlusion

Visibility caches have been shown that can be used to compute noise-free ambient occlusion with a small fraction of computation [14]. In our implementation, we trace 1024 shadow rays generated by the equal-area spherical mapping [32] to compute the ambient occlusion value at cache locations. Figure 6 demonstrates the equal-sample comparisons of various cache distributions. We use 50K caches for Statues and 120K caches for Bathroom, Conference, and Kitchen. Disregarding scene geometry, **IM** produces the largest error in all test cases. **SD** and **SD+AS** outperform **IM**

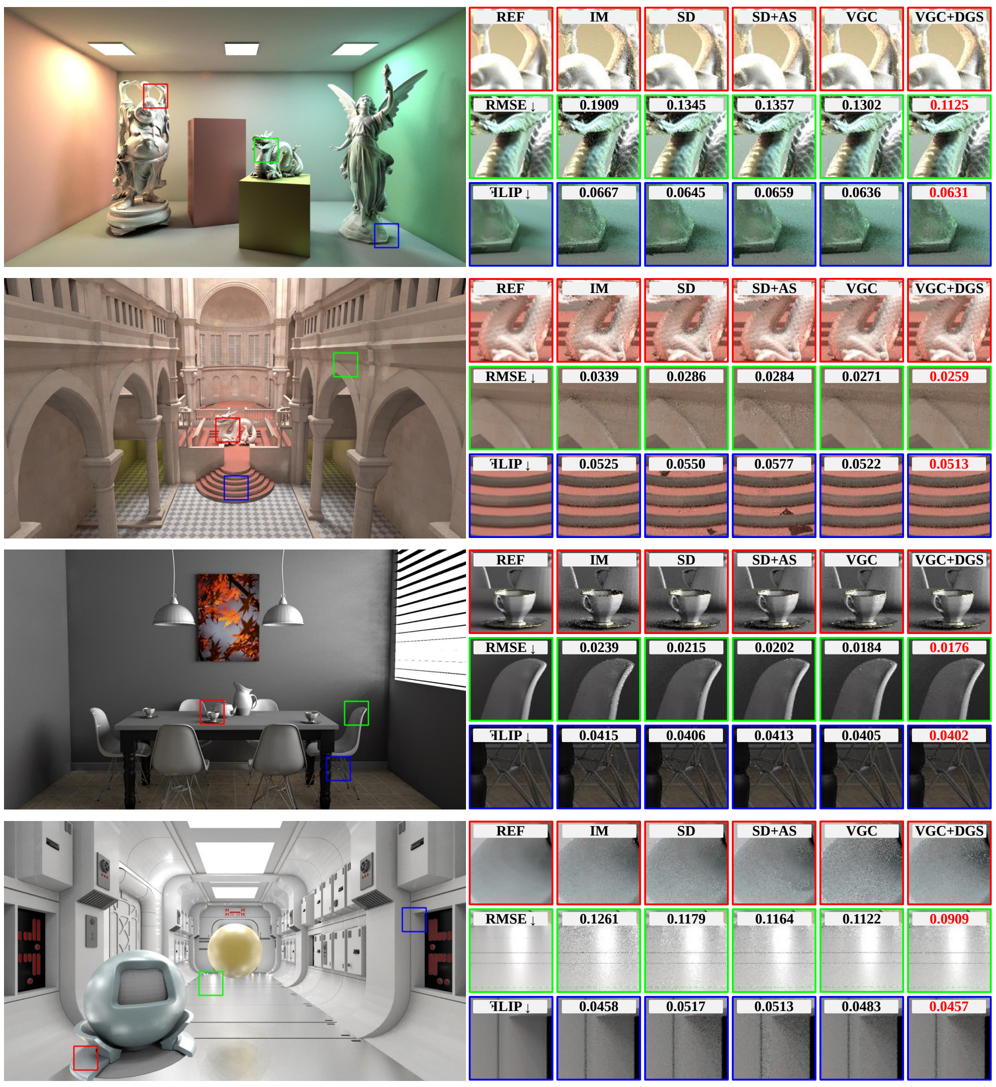


Fig. 5: Equal-cache comparison (25K caches) of importance caching on CornellBox, Sibenik, DiningRoom and Robots with different cache distributions: image-space caching (**IM**), spatial-directional BVH (**SD**) [17], spatial-directional BVH with adaptive sampling (**SD+AS**) [16], and our method (**VGC** only and **VGC+DGS**). The full images on the left are rendered with our full method **VGC+DGS**. The proposed method offers a more accurate sampling function estimation and outperforms other approaches both visually and numerically.

for taking geometry into consideration. However, they tend to spend too much effort on curved surfaces, leading to insufficient cache density in the spatial domain. **VGC** shows great improvement over **SD** in Bathroom and Conference. Nonetheless, it

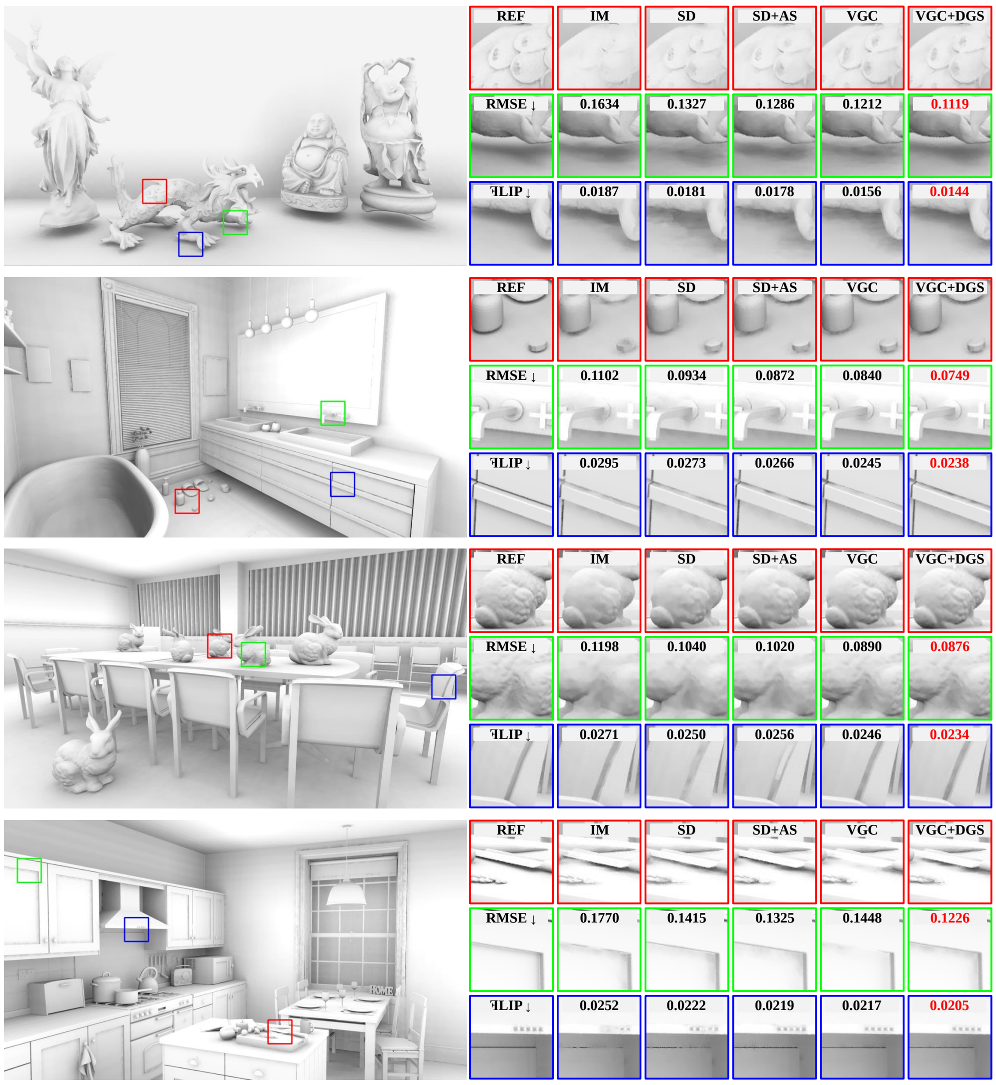


Fig. 6: Equal-cache comparison of cache-based ambient occlusion on Statues, Bathroom, Conference, and Kitchen with different cache distributions: image-space uniform sampling (**IM**), spatial-directional BVH (**SD**) [17], spatial-directional BVH with adaptive sampling (**SD+AS**) [16], and our method (**VGC** only and **VGC+DGS**). The **Statues** uses 50K caches while others use 120K caches. The full images on the left are rendered with our **VGC+DGS**. Our method produces the most visually pleasing results and the smallest RMSE and ꟻLIP errors in all cases. Our discontinuity-guided sampling improves the results in regions of shadow and small geometry details.

Scene	Bathroom (AO)		Conference (IrC)		DiningRoom (ImC)	
Method/Error	RMSE↓	ꟻLIP↓	RMSE↓	ꟻLIP↓	RMSE↓	ꟻLIP↓
VGC-only	0.0840	0.0245	0.0031	0.0602	0.0184	0.0405
VGC+AS	0.0811	0.0252	0.0032	0.0625	0.0182	0.0411
DGS (diff)	0.0801	0.0248	0.0030	0.0596	0.0176	0.0402
DGS (geom)	0.0770	0.0245	0.0030	0.0599	0.0189	0.0410
DGS (full)	0.0749	0.0238	0.0029	0.0594	0.0184	0.0405

Table 2: Comparisons of using various components in the discontinuity-guided sampling for ambient occlusion (AO), irradiance caching (IrC), and importance caching (ImC). **DGS (diff)** and **DGS (geom)** stand for the settings using the difference between the application-dependent samples and the geometry term for constructing the discontinuity map, respectively. **DGS (all)** uses both terms. We also compare with the configuration that combines our view-guided sampling with adaptive sampling [16] (**VGC+AS**).

struggles to capture the occlusion changes due to fine geometry, resulting in larger errors on the cabinets of Kitchen. This issue is addressed by introducing **DGS**, which improves the image quality on geometry details, such as the cabinets in the Kitchen and Bathroom. In Statues, **SD**, **SD+AC**, and **VGC** all produce artifacts near the shadows on the ground because they cannot capture the shading change on simple geometry. Our full method reduces the artifacts of shadows and produces the most visually pleasing result.

4.4 Ablation Studies

Various components v.s. accuracy. In Table 2, we present the performances of various components used in discontinuity-guided sampling. The setting that only uses the difference between the application-dependent samples for constructing the discontinuity map is denoted by **DGS (diff)**, while **DGS (geom)** uses the geometry term only. The full method that combines both terms is denoted by **DGS (full)**. To compare the effectiveness of discontinuity-guided sampling with adaptive sampling [16], we also include a configuration that combines our view-guided clustering and adaptive sampling denoted by **VGC+AS**. For irradiance caching and ambient occlusion, the experiment results demonstrate that the difference between the application-dependent samples and the geometry term is crucial for various scenes. The geometry term plays an essential role in scenes containing rich small geometry features such as Bathroom, while the difference term has a more significant impact in scenes with many flat surfaces such as Conference. Our full method effectively combines the strength of both terms. We discovered that including the geometry term would not lead to better results for importance caching. One possible reason is that the cached values are used as sampling functions instead of direct interpolation for a pixel value. This implies that the advantage of incorporating the geometry term is diminished. In all applications, view-guided clustering combined with discontinuity-guided sampling outperforms the one combined with adaptive sampling (**VGC+AS**).

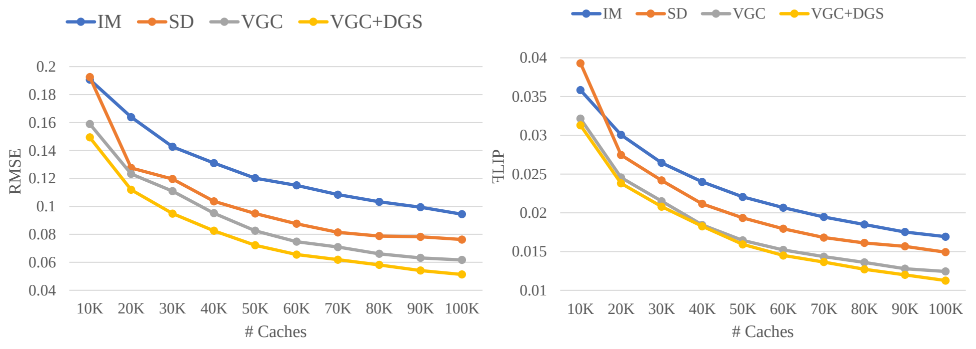


Fig. 7: Relationship between the number of caches and RMSE and ꟻLIP on Staircase scene (Figure 1) rendered with ambient occlusion. The plots showcase that our method is consistently more efficient than other methods.

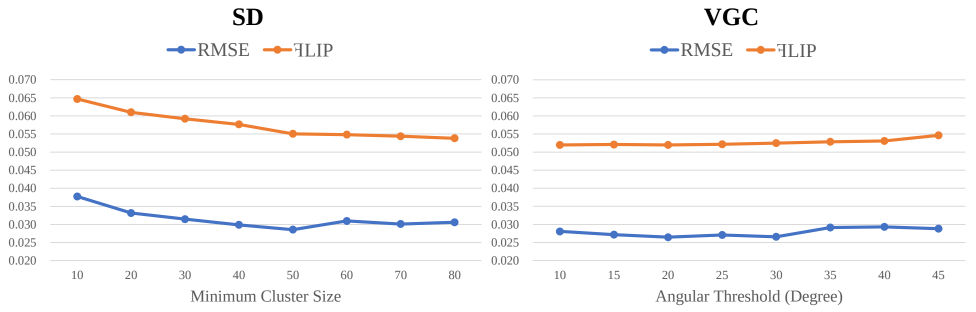


Fig. 8: Relationship between the parameter selection and RMSE and ꟻLIP on Sibenik scene rendered with importance caching. We adjust the minimum cluster size for **SD** and the angular threshold for **VGC**. The plots showcase that our method is less sensitive to the selection of parameters.

Number of caches v.s. accuracy. Figure 7 demonstrates the relationship between the number of caches and different metrics (RMSE and ꟻLIP) on Staircase rendered with ambient occlusion. Our method **VGC** consistently produces superior results than previous methods, ranging from preview quality using 10K caches to high-quality rendering using 100K caches. Furthermore, the figure highlights that our two-stage method (**VGC+DGS**) reduces numerical errors by utilizing partial rendering results.
Parameters v.s. accuracy. We conducted an experiment to investigate how various parameters affect the numerical errors rendered with **SD** and our **VGC** using 25K importance caches on Sibenik scene. For **SD**, we adjust the minimum cluster size which controls the relative importance between world-space geometric proximities and image-space importance. For **VGC**, we altered the angular threshold, which influences the angular and spatial proximity. Using a smaller angular threshold results in more caches allocated on curved surfaces, at the expense of sparser samples in the spatial extent. The results presented in Figure 8 indicate that **VGC** consistently produces

	ratio	40%	30%	20%	10%	0% (**VGC**)
Staircase	RMSE↓	0.0596	0.0584	0.0578	0.0587	0.0661
(AO, 80K caches)	ꟻLIP↓	0.0135	0.0133	0.0129	0.0130	0.0136
DiningRoom	RMSE↓	0.0192	0.0187	0.0188	0.0175	0.0184
(ImC, 25K caches)	ꟻLIP↓	0.0434	0.0423	0.0417	0.0400	0.0405

Table 3: Comparisons of the various fractions of total cache budgets moved from **VGC** to **DGS**. The last column (0%) means using **VGC** only. According to the experiment results, we suggest using a ratio between 10% to 20%.

Scene	CrytekSponza (irradiance caching)			Sibenik (importance caching)			Bathroom (ambient occlusion)		
Number of caches	40K			25K			120K		
Method	**IM**	**SD**	**Ours**	**IM**	**SD**	**Ours**	**IM**	**SD**	**Ours**
Shading points gen.	0.42	0.42	0.42	0.40	0.40	0.40	0.35	0.35	0.35
Initial cache gen.	0.14	0.53	0.42	0.14	0.50	0.43	0.14	0.58	0.46
Initial cache comp.	349.90	350.08	297.22	25.40	25.71	21.93	5.04	5.05	4.17
Adaptive cache gen.			0.25			0.25			0.26
Adaptive cache comp.			53.74			3.87			0.80
KD-Tree building	0.01	0.01	0.01	0.01	0.01	0.01	0.04	0.04	0.04
Total cache gen.	350.47	351.04	352.06	25.95	26.62	26.89	5.57	6.02	6.08
Rendering	7.30	7.31	7.24	16.52	16.14	16.28	5.80	5.76	5.70

Table 4: Time (in seconds) profiling of each step in various methods. The initial cache computation time of our method is shorter than the other single-pass methods (**IM** and **SD**) because we redistribute 15% cache budget to the second (adaptive) stage. The adaptive cache generation time of our method includes the time to build the discontinuity map and draw adaptive samples using importance sampling.

better results than **SD** over a wide range of parameters. Furthermore, **VGC** is less sensitive to parameter selection than **SD**, as it exhibits less fluctuation in the results.
Sample budgets for VGC and DGS. To investigate the appropriate sample budgets for **VGC** and **DGS**, we render the ambient occlusion of the Staircase scene with 80K caches and compute global illumination of the DiningRoom scene with 25K importance caches, using various ratios of samples moved from **VGC** to **DGS**. The results are presented in Table 3. Although the discontinuity-guided sampling pass does improve the results compared to using **VGC**-only (0% in the table), it relies on the information from the first stage (*i.e.*, **VGC**). Therefore, an excessively high allocation of samples to **DGS** could lead to sub-optimal results. Our experiment suggests that the best ratio for the sample budget is between 10% to 20%.

4.5 Performance Profiling

Table 4 demonstrates the detailed timing of each step of the various methods on CrytekSponza (irradiance caching), Sibenik (importance caching), and Bathroom (ambient occlusion). Compared to **SD**, our method can produce better results with negligible overheads in all applications. Compared to **IM**, the overheads to improve the cache distributions in importance caching and ambient occlusion are slightly larger than in irradiance caching because the computation of cache data is less expensive. However, considering the visual quality improvements over **IM** in Figure 5 and Figure 6, and the numerical error improvements demonstrated in the ablation study: number of caches v.s. accuracy (Figure 7), it is worth improving the cache distribution with the overhead.

4.6 Limitations and Future Work

Our discontinuity-guided sampling method is effective, but it combines cache differences and small geometry features using a heuristic approach. As a result, a large geometry term might lead to oversampling of regions where the differences between caches are already small. Although we can introduce a hyperparameter to emphasize the relative importance of the two terms, this control is not straightforward. Therefore, we are interested in investigating whether a data-driven approach utilizing the computation results obtained in the first stage can improve the performance.

Another interesting future direction is to extend our method to the temporal domain for rendering animations. We are planning to design criteria for reusing cache across frames. Finally, we would also like to develop a GPU version of the proposed method for real-time rendering.

5 Conclusion

In this paper we propose a new algorithm to improve the sample distribution for cache-based rendering. Our method comprises two stages. In the first stage, an initial cache distribution is determined based on a new shading point clustering algorithm called view-guided clustering. The new algorithm is more robust for striking a better balance between geometric proximities and image-space importance. It is more intuitive in parameter settings than in previous methods. Our second stage further improves the cache distribution by utilizing the computation results at the initial caches. It allocates additional caches in regions with complex shading. The result is an efficient algorithm for determining cache locations for cache-based techniques. Experiments show the proposed method offers significant improvements over other strategies for generating caches.

Acknowledgments. We thank the anonymous reviewers for their valuable comments, and the creators or providers of the models and textures used in this paper: Staircase, CornellBox, DiningRoom, Bathroom, and Kitchen, via Benedikt Bitterli's rendering resources [34]; CrytekSponza, Conference, and Sibenik from McGuire Computer Graphics Archive [35]; This work was supported in part by the National Science and Technology Council (NSTC) under grants 111-2222-E-305-001-MY2 and JSPS Grant-in-Aid JP23K16921.

6 Compliance with Ethical Standards

6.1 Potential Conflicts of Interest

The following are potential conflicts of interest:

- Yung-Yu Chuang (https://www.csie.ntu.edu.tw/~cyy/)
- Tzu-Mao Li (https://cseweb.ucsd.edu/~tzli/)

6.2 Research Involving Human Participants and/or Animals

This research did not involve human participants.

7 Data Availability Statement

The source code and test scenes used in this work are available from the corresponding author on reasonable request.

References

[1] Ward, G.J., Rubinstein, F.M., Clear, R.D.: A ray tracing solution for diffuse interreflection. In: Proc. SIGGRAPH, pp. 85–92 (1988). https://doi.org/10.1145/378456.378490

[2] Ward, G., Heckbert, P.: Irradiance gradients. In: Proc. Eurographics Workshop on Rendering, pp. 85–98 (1992). https://doi.org/10.1145/1401132.1401225

[3] Tabellion, E., Lamorlette, A.: An approximate global illumination system for computer generated films. ACM Trans. Graph. (Proc. SIGGRAPH) **23**(3), 469–476 (2004) https://doi.org/10.1145/1186562.1015748

[4] Gautron, P., Kriválek, J., Bouatouch, K., Pattanaik, S.: Radiance Cache Splatting: A GPU-Friendly Global Illumination Algorithm. In: Proc. Eurographics Symposium on Rendering (2005). https://doi.org/10.1145/1187112.1187154

[5] Brouillat, J., Gautron, P., Bouatouch, K.: Photon-driven irradiance cache. Computer Graphics Forum **27**(7), 1971–1978 (2008) https://doi.org/10.1111/j.1467-8659.2008.01346.x

[6] Schwarzhaupt, J., Jensen, H.W., Jarosz, W.: Practical hessian-based error control for irradiance caching. ACM Trans. Graph. (Proc. SIGGRAPH) **31**(6) (2012) https://doi.org/10.1145/2366145.2366212

[7] Křivánek, J., Gautron, P., Pattanaik, S., Bouatouch, K.: Radiance caching for efficient global illumination computation. IEEE Trans. on Visualization and Computer Graphics **11**(5), 550–561 (2005) https://doi.org/10.1109/TVCG.2005.83

[8] Křivánek, J., Bouatouch, K., Pattanaik, S., Žára, J.: Making radiance and irradiance caching practical: Adaptive caching and neighbor clamping. In: Proc. Eurographics Symposium on Rendering, pp. 127–138 (2006). https://doi.org/10.2312/EGWR/EGSR06/127-138

[9] Gassenbauer, V., Krivanek, J., Bouatouch, K.: Spatial Directional Radiance Caching. Computer Graphics Forum (2009) https://doi.org/10.1111/j.1467-8659.2009.01496.x

[10] Scherzer, D., Nguyen, C.H., Ritschel, T., Seidel, H.-P.: Pre-convolved radiance caching. Computer Graphics Forum **31**(4), 1391–1397 (2012) https://doi.org/10.1111/j.1467-8659.2012.03134.x

[11] Rehfeld, H., Zirr, T., Dachsbacher, C.: Clustered pre-convolved radiance caching. In: Proc. Eurographics Symposium on Parallel Graphics and Visualization, pp. 25–32 (2014). https://doi.org/10.2312/pgv.20141081

[12] Zhao, Y., Belcour, L., Nowrouzezahrai, D.: View-dependent radiance caching. In: Proc. Graphics Interface (2019). https://doi.org/10.20380/GI2019.22

[13] Müller, T., Rousselle, F., Novák, J., Keller, A.: Real-time neural radiance caching for path tracing. ACM Trans. Graph. (Proc. SIGGRAPH) **40**(4) (2021) https://doi.org/10.1145/3450626.3459812

[14] Clarberg, P., Akenine-Moeller, T.: Exploiting visibility correlation in direct illumination. In: Computer Graphics Forum (Proc. EGSR), pp. 1125–1136 (2008). https://doi.org/10.1111/j.1467-8659.2008.01250.x

[15] Georgiev, I., Křivánek, J., Popov, S., Slusallek, P.: Importance caching for complex illumination. Computer Graphics Forum (Proc. Eurographics) **31**(2), 701–710 (2012) https://doi.org/10.1111/j.1467-8659.2012.03049.x

[16] Yoshida, H., Nabata, K., Iwasaki, K., Dobashi, Y., Nishita, T.: Adaptive importance caching for many-light rendering. Journal of WSCG **23**(1), 65–71 (2015)

[17] Ou, J., Pellacini, F.: LightSlice: Matrix slice sampling for the many-lights problem. ACM Trans. Graph. (Proc. SIGGRAPH Asia) **30**(6) (2011) https://doi.org/10.1145/2070781.2024213

[18] Wu, Y.-T., Chuang, Y.-Y.: VisibilityCluster: Average directional visibility for many-light rendering. IEEE Trans. on Visualization and Computer Graphics **19**(9), 1566–1578 (2013) https://doi.org/10.1109/TVCG.2013.21

[19] Wu, Y.-T., Li, T.-M., Lin, Y.-H., Chuang, Y.-Y.: Dual-matrix sampling for scalable translucent material rendering. IEEE Transactions on Visualization and Computer Graphics **21**(3), 363–374 (2015) https://doi.org/10.1109/TVCG.2014.2385059

[20] Vévoda, P., Kondapaneni, I., Křivánek, J.: Bayesian online regression for adaptive direct illumination sampling. ACM Trans. Graph. (Proc. SIGGRAPH) **37**(4) (2018) https://doi.org/10.1145/3197517.3201340

[21] Andersson, P., Nilsson, J., Shirley, P., Akenine-Möller, T.: Visualizing Errors in Rendered High Dynamic Range Images. In: Eurographics Short Papers (2021). https://doi.org/10.2312/egs.20211015

[22] Keller, A.: Instant radiosity. In: Proc. SIGGRAPH, pp. 49–56 (1997). https://doi.org/10.1145/258734.258769

[23] Jensen, H.W.: Global illumination using photon maps. In: Rendering Techniques, pp. 21–30 (1996). https://doi.org/10.1007/978-3-7091-7484-5_3

[24] Silvennoinen, A., Lehtinen, J.: Real-time global illumination by precomputed local reconstruction from sparse radiance probes. ACM Trans. Graph. **36**(6) (2017) https://doi.org/10.1145/3130800.3130852

[25] Jarosz, W., Donner, C., Zwicker, M., Jensen, H.W.: Radiance caching for participating media. ACM Trans. Graph. **27**(1) (2008) https://doi.org/10.1145/1330511.1330518

[26] Marco, J., Jarabo, A., Jarosz, W., Gutierrez, D.: Second-order occlusion-aware volumetric radiance caching. ACM Trans. Graph. **37**(2) (2018) https://doi.org/10.1145/3185225

[27] Dubouchet, R.A., Belcour, L., Nowrouzezahrai, D.: Frequency based radiance cache for rendering animations. In: Computer Graphics Forum (Proc. EGSR) (2017). https://doi.org/10.2312/sre.20171193

[28] Patry, J., Wright, D., Halen, H., Hayward, K., Brinck, A., Bei, X.: Advances in Real-Time Rendering in Games, Part 2. SIGGRAPH Course (2021). https://advances.realtimerendering.com/s2021/

[29] McLaren, J.: The Technology of The Tomorrow Children. Game Developers Conference (GDC) (2015). https://www.gdcvault.com/play/1022428/The-Technology-of-The-Tomorrow

[30] Bitterli, B., Wyman, C., Pharr, M., Shirley, P., Lefohn, A., Jarosz, W.: Spatiotemporal reservoir resampling for real-time ray tracing with dynamic direct lighting. ACM Trans. Graph. (Proc. SIGGRAPH) **39**(4) (2020) https://doi.org/10.1145/3386569.3392481

[31] Lin, D., Kettunen, M., Bitterli, B., Pantaleoni, J., Yuksel, C., Wyman, C.: Generalized resampled importance sampling: Foundations of restir. ACM Trans. Graph. (Proc. SIGGRAPH) **41**(4) (2022) https://doi.org/10.1145/3528223.3530158

[32] Clarberg, P.: Fast Equal-Area Mapping of the (Hemi)Sphere using SIMD. Journal of Graphics Tools **13**(3), 53–68 (2008) https://doi.org/10.1080/2151237X.2008.10129263

[33] Pharr, M., Jakob, W., Humphreys, G.: Physically Based Rendering: From Theory to Implementation, 3rd edn., p. 1266. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA (2016)

[34] Bitterli, B.: Rendering resources. https://benedikt-bitterli.me/resources/ (2016)

[35] McGuire, M.: Computer Graphics Archive. https://casual-effects.com/data