CrossMark

Non-redundant rendering for efficient multi-view scene discretization

Naiwen Xie[1] · Lili Wang[1] · Voicu Popescu[2]

Abstract A powerful approach for managing scene complexity is to sample the scene with a set of images. However, conventional images from nearby viewpoints have a high level of redundancy, which reduces scene sampling efficiency. We present non-redundant rendering, which detects and avoids redundant samples as the image is computed. We show that non-redundant rendering leads to improved scene sampling quality according to several view-independent and view-dependent metrics, compared to conventional scene discretization using redundant images and compared to depth peeling. Non-redundant images have a higher degree of fragmentation and, therefore, conventional approaches for scene reconstruction from samples are ineffective. We present a novel reconstruction approach that is well suited to scene discretization by non-redundant rendering. Finally, we apply non-redundant rendering and scene reconstruction techniques to soft shadow rendering where we show that our approach has an accuracy advantage over conventional images and over depth peeling.

Keywords Scene sampling · Sampling redundancy · Non-redundant sampling

✉ Naiwen Xie
    sienaiwun@gmail.com

1   Beihang University, Beijing, China

2   Purdue University, West Lafayette, USA

1 Introduction

Graphics applications where interactivity is an essential concern cannot handle the scene geometry at its full complexity. Examples include remote visualization of a large scene on a thin client (e.g., a smartphone), and rendering expensive effects such as soft shadows, reflections, or ambient occlusion. Despite decades of research, geometry simplification remains largely an open research problem. Remaining challenges include meeting a polygon budget while bounding the simplification error, supporting dynamic scenes, and providing a smooth transition between geometry levels of detail.

Approximating geometry through sampling has the potential to overcome these challenges by providing a fine grain control of the approximation error and by providing real-time performance suitable for dynamic scenes. An image with per pixel depth is a powerful method for approximating scene geometry. Such a depth image can be rendered quickly with the help of graphics hardware, and the cost of processing the scene is amortized over a large number of samples.

However, a conventional depth image does not capture enough scene samples to adequately support graphics applications. For example, in the case of remote visualization, any viewpoint translation at the client exposes parts of the scene that were not captured by the depth image, which leads to highly objectionable artifacts which we call occlusion errors. In the case of soft shadows, a conventional shadow map does not capture all surfaces visible from an area light source, and penumbra regions cannot be shaded correctly.

The problem with a conventional image is that it has a single viewpoint. A straightforward solution to the single viewpoint limitation is to rely on multiple images. However, multiple images are highly redundant. One prior solution to the redundancy problem is to eliminate the redundant samples a posteriori, i.e., once the images were rendered. Another

⌖ Springer

solution is *depth peeling*, where the scene is rendered multiple times from the same viewpoint, each time z-buffering beyond the previous layer. The problem with depth peeling is that it renders the scene from the same viewpoint, which limits sampling quality. For example, when the viewpoint belongs to the plane of a wall in a scene, the wall will not be sampled no matter how many layers are peeled away.

We introduce *non-redundant rendering*, a novel approach for sampling a 3D scene with multiple images while avoiding redundancy. Non-redundant rendering is based on a very simple idea: a sample is kept only if it has not already been acquired. As the second and subsequent images are rendered, a candidate sample is kept only if it is not redundant with samples already acquired by the previous images. Unlike for the prior art approach of discarding redundancy a posteriori, which does not replace a redundant sample with a new sample, in the case of non-redundant rendering, redundancy is avoided a priori, so the redundant sample does not prevent gathering a new, non-redundant sample. The result is higher sampling efficiency. Compared to the prior approach of depth peeling, non-redundant rendering samples the scene from multiple viewpoints, resulting in better sampling quality.

We demonstrate the higher sampling efficiency of non-redundant rendering compared to conventional sampling with multiple images, and compared to depth peeling. When a scene is discretized with three conventional depth images, the second and third images have redundancy rates of 70 and 80 %, respectively. When non-redundant images are used, the 62 and 76 % of the samples of the second and third images are samples not captured by the conventional depth images. Out of these disoccluded samples, 89 and 69 % are useful since they are visible from an intermediate viewpoint. When the scene is discretized with three-layer depth peeling, the second and third layers disocclude fewer samples (47 and 52 %) and fewer of them are useful (73 and 71 %), compared to the non-redundant discretization.

Most applications of scene discretization require reconstructing the scene from the sample-based representation. For example, the remote visualization application requires rendering the scene from novel viewpoints at the client from the samples transmitted from the server. In soft-shadow rendering, one has to compute approximate shadow maps from the scene discretization, for each of the light samples. Scene reconstruction from samples has been studied extensively in the context of point-based rendering or in the context of modeling from acquired point clouds. Non-redundant rendering poses the challenge of a higher degree of fragmentation. Replacing the redundant samples with new samples creates additional depth discontinuities which are challenging for prior reconstruction approaches, which underestimate geometry at surface edges. We present a reconstruction method that can handle the fragmentation of non-redundant images. The

method provides a better geometry approximation at surface edges, while the reconstruction inside the surface is watertight and with no overdraw.

We demonstrate our non-redundant rendering and scene reconstruction methods in the context of soft shadow rendering. In Fig. 1, an area light source is sampled at 16×16 resolution, a shadow map is rendered for each light sample, and the shadow maps are used to compute the soft shadows. For the conventional and the non-redundant scene discretizations, the shadow maps are computed from four depth images placed at the corners of the light source. For depth peeling, the shadow maps are computed from four layers rendered from the center of the light. Our scene discretization produces a better approximation of the light sample shadow maps, resulting in a smaller average shadow intensity error. We refer the reader to the accompanying video.

2 Prior work

The idea of using an image as a simplified representation of scene geometry is used in billboard rendering [8]. The construction of a billboard is inexpensive and the billboard provides a good approximation when seen from a distance. Moreover, a billboard can be intersected inexpensively with a single ray, which supports higher order rendering effects such as specular reflections. However, modeling fidelity decreases when the observer moves closer to the billboard.

A depth image [12] greatly increases modeling fidelity by modulating the depth of the base plane with thousands of values. Constructing a depth image has the same low cost of constructing a billboard. Intersecting a depth image with a single ray is more expensive than in the case of a billboard, but it is still much faster than intersecting the original scene geometry: a depth image is intersected efficiently with a ray by tracing the ray's projection onto the image.

However, a single depth image might not capture all the geometry needed by the application. For example, in the case of remote visualization, using a single depth image at the client creates disocclusion errors for the slightest viewpoint translation. In the case of specular reflection rendering, a reflected ray might intersect the scene geometry at a point that is not visible in the depth image.

The simplest idea for eliminating disocclusion errors is to use additional depth images [9]; however, multiple depth images are redundant. The higher the number of depth images, the higher is the redundancy. Any new depth image brings fewer and fewer new samples. Layered representations such as the LDI (layered depth image) [18] and LDI trees [4] provide scene sampling at multiple scales which supports scene geometry approximations with multiple levels of detail. LDIs are constructed by first rendering multiple depth images and then by combining
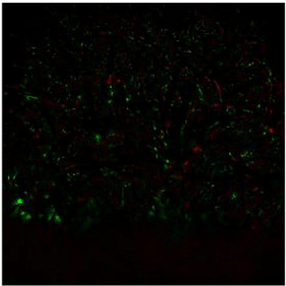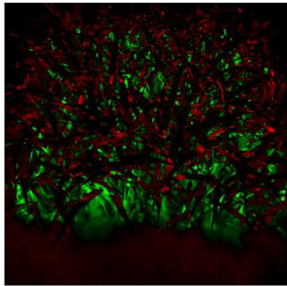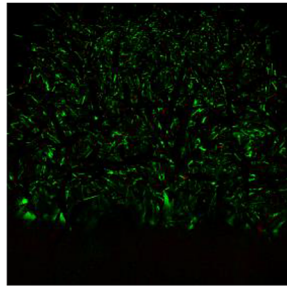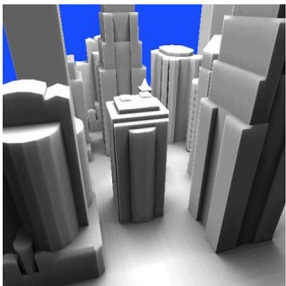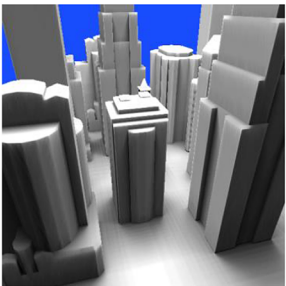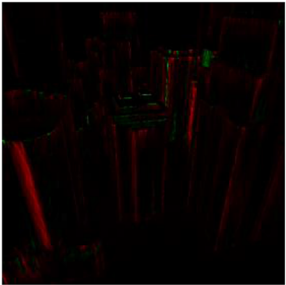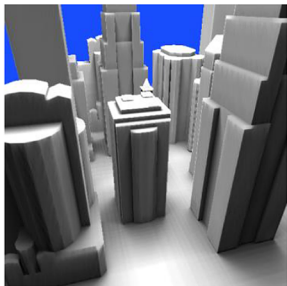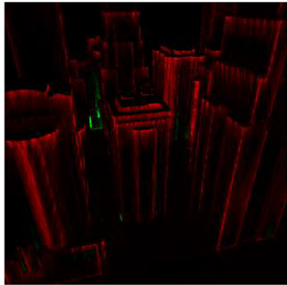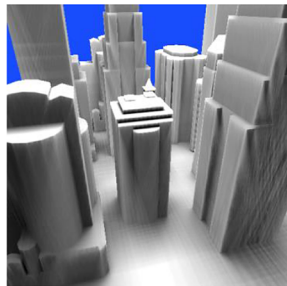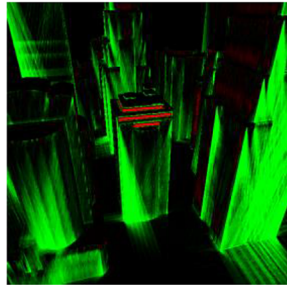
Geometry (Ground Truth)	Non-Redundant Rendering	Conventional Sampling	Depth peeling

Fig. 1 Soft shadows rendered using the original geometry and using three scene discretizations. Our non-redundant scene discretization produces the most accurate shadows

the depth images to remove the redundant samples. LDIs can be used in rendering indirect illumination and soft shadows [10,19]. One disadvantage is the two-step construction that renders a set of conventional images first and then removes the redundancy among the images in the set. A second disadvantage is that the LDI construction only removes redundant samples and does not replace them with new samples. Finally, sample connectivity is difficult to infer for LDIs, which complicates reconstruction.

Incremental textured depth meshes [20] is a technique for discretizing a complex scene that focuses on determining a good set of viewpoints from where to sample. This issue is complementary to the issue of improving the sampling capability of an individual depth image, given a viewpoint, which is the focus of our work. Textured depth meshes offer the possibility of rendering with a predetermined geometry budget, or with a predetermined fidelity target. In other words, our non-redundant sampling approach can be used in conjunction with the viewpoint optimization proposed in the earlier work. Furthermore, in the textured depth meshes approach, redundancy is detected after the fact. Like for LDIs, the redundant samples are removed, but they are not replaced with non-redundant ones. Our method precludes the computation of redundant samples, which do not get a chance to occlude a potentially useful, non-redundant sample. Finally, we demonstrate our sampling capability in the context of soft shadow rendering, and not just in the context of walk-through acceleration.

Graphic artists have known for a long time that relaxing the single viewpoint constraint can be used to achieve effects that strengthen artistic expression. The idea is used in multiperspective rendering where the image is generalized to integrate samples from multiple viewpoints. Examples include multiple center of projection images [15], street panoramas [1,16,17], general linear camera images [22], and occlusion camera images [13]. All these multiperspective images have better sampling capability than conventional images, but constructing multiperspective images that capture sufficient samples in the case of a complex scene remains challenging.

Depth peeling [7,11] samples a scene non-redundantly in multiple passes, with each pass going beyond the layer acquired by the previous pass. Like in the case of LDIs, depth peeling generates an image with deep pixels that store a variable number of samples. Depth peeling is useful when rendering transparency, and also when rendering complex opaque geometry, such as trees [11]. The method is fast as it simply requires consulting the z-buffer of the previous pass to avoid redundancy. Dual depth peeling [6] leverages the GPU's min-max depth buffer to capture both the nearest and the farthest layers in each pass, improving peeling efficiency. Multilayer depth peeling [2,3] relies on Multiple Render Targets (MRTs) to further improve the efficiency of each pass. A fundamental shortcoming of depth peeling is that each pass samples the scene from the same viewpoint, which has two disadvantages. First, surfaces that are seen at an acute angle from the viewpoint will be sampled poorly no matter how many layers are peeled away. Second, peeling layers away X-rays the scene, and many of the samples acquired might not be visible from any viewpoint in the neighborhood of the reference viewpoint. In other words, the samples recovered by depth peeling might not be useful to the application.

Our non-redundant rendering method combines the sampling advantages of multiple viewpoints with the depth peeling advantage of avoiding sample redundancy on the fly, as the images are rendered. As shown in this paper, non-redundant rendering leads to better sampling quality compared to traditional multiple image or depth peeling sampling.

Scene discretization techniques, like the one proposed by our work, and geometry simplification techniques share the goal of computing a lightweight scene representation that can be used to accelerate graphics applications. The goal is pursued from opposite directions: scene discretization adds samples until the set of samples is satisfactory, and geometry simplification reduces the number of geometric primitives until a geometry budget is met. Some geometric simplification approaches evaluate the approximation error in image space [5], and the solutions developed for finding the set of viewpoints from where to evaluate the error can be used in the context of finding the set of viewpoints from where to discretize the scene.

Our paper is also related to the body of work on visibility computation. For example, the guided visibility sampling approach [21] chooses the rays along which visibility is probed based on the results of earlier probes, which is similar in spirit to our approach of extending the ray beyond the currently encountered sample should that sample be already acquired by the previous images. However, in our case visibility rays are grouped in images which enables probing visibility in feed-forward fashion with a small per-ray amortized cost. Moreover, for complex scenes the potentially visible set computed by visibility methods could be very large, requiring simplification in an additional step.

3 Non-redundant rendering

Non-redundant rendering is an approach for sampling a scene from different viewpoints using non-redundant images. We define redundancy in one of two ways, and then we describe the non-redundant rendering algorithm.

3.1 Redundancy definition

Consider a scene S modeled with triangles, and two depth images I_0 and I_1 that render S from viewpoints V_0 and V_1. Given a sample s_1 in I_1, we want to examine the question whether s_1 is redundant with the samples acquired by I_0. We define multiple types of redundancy.

Strict redundancy Sample s_1 is strictly redundant with I_0 iff there is a sample s_0 in I_0 where s_1 and s_0 acquire the same point on the same scene triangle. Such redundancy (almost) never happens since a triangle is (almost) never sampled at the
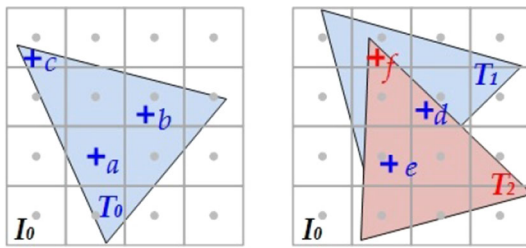
Fig. 2 Illustration of various redundancy scenarios for a sample based on possible reprojection locations

same point. We do not use strict redundancy in non-redundant rendering.

Visibility redundancy Sample s_1 is visibility redundant with I_0 iff s_1 is visible from the viewpoint V_0 of I_0. This definition of redundancy makes abstraction of the resolution of I_0. In other words, if I_0 had infinite resolution, I_0 would have a sample s_0 that is strictly redundant with s_1.

Pixel redundancy Sample s_1 is pixel redundant with I_0 iff s_1 projects onto I_0 at a pixel where I_0 captures a sample s_0 of the same triangle as s_1. This definition of redundancy does not require that I_0 has exactly the same sample, but just that it has a sample of the same triangle acquired from within the same pixel.

Figure 2 shows image I_0. The vertical crosses indicate possible I_0 reprojections of s_1 to illustrate multiple redundancy scenarios. In scenarios a, b, and c in Fig. 2 left, it is assumed that s_1 samples triangle T_0 in I_1. If s_1 reprojects at a, s_1 is strictly redundant with I_0 since s_1 reprojects exactly at the center of a pixel of I_0. If s_1 reprojects at b, s_1 is both visibility and pixel redundant with I_0, because s_1 is visible from the viewpoint V_0 of I_0, and because b is at a pixel where I_0 samples the same triangle T_0 that s_1 samples in I_1. If s_1 reprojects at c, s_1 is visibility redundant with I_0 because there is a direct line of sight from V_0 to s_1. However, s_1 is not pixel redundant with I_0 because I_0 captures a different triangle T_1 at the pixel that contains c. For scenario d in Fig. 2 right, it is assumed that s_1 samples T_1 in I_1. In this scenario, s_1 is not visibility redundant, since s_1 is occluded by T_2 from V_0 (i.e., in I_0). However, s_1 is pixel redundant with I_0 because I_0 captures the same triangle T_1 at the pixel that contains d. In scenario e, s_1 samples T_1 in I_1 and s_1 is not visibility and not pixel redundant with I_0. In scenario f, where it is assumed that s_1 samples T_2 in I_1, s_1 is visibility redundant but not pixel redundant with I_0.

3.2 Non-redundant rendering algorithm

Given a scene S modeled with triangles and given a set of n views defined by planar pinhole cameras PPC_i, we render n non-redundant depth images I_i with Algorithm 1.

Algorithm 1 Non-redundantRendering

Input: scene S, PPC_i ($i = 0$ to $n - 1$).
Output: non-redundant depth images I_i ($i = 0$ to $n - 1$).
1: **for each** image I_i ($i = 0$ to $n - 1$) **do**
2:    initialize the z-buffer of I_i to far
3:    **for each** triangle t in S **do**
4:       project t with PPC_i to t'
5:       **for all** pixels p covered by t' **do**
6:          compute sample s of t at p
7:          **if** s fails the z-buffer test **then**
8:             **continue**
9:          **for all** previous images I_j ($j < i$) **do**
10:            **if** s is redundant with I_j **then**
11:               mark s as redundant
12:               **break**;
13:         **if** s is not redundant **then** write s in I_i

---

Algorithm 2 VisibilityRedundancyTest(s, PPC_i, V_j, S)

Input: sample s, view PPC_i of image where s is generated, viewpoint V_j of image I_j where redundancy is tested, and scene S.
Output: visibility redundancy of s with I_j
1: Unproject s to 3D point a using PPC_i
2: Raytrace $b == V_j a \cap S$
3: **return** ($b == a$)

Algorithm 3 PixelRedundancyTest(s, t, PPC_i, I_j, PPC_j)

Input: sample s, id t of triangle sampled by s in the image I_i where s is generated, view PPC_i of I_i, image I_j where redundancy is tested, and view PPC_j of I_j.
Output: pixel redundancy of s with I_j
1: Reproject s to pixel r in I_j using PPC_i and PPC_j
2: let t_r be the id of the triangle sampled by I_j at r
3: **return**($t == t_r$)

The images are rendered one at the time. Each image is rendered by taking a pass over the scene geometry. Each triangle is projected and rasterized conventionally to generate samples. A sample is kept only if it passes the conventional z-buffer test, and if it is not redundant with any of the previously rendered images. The first image (i.e., I_0) is a conventional image since there are no previous images against which to check for redundancy. The following images will avoid redundant samples. The redundancy check is implemented in one of two ways to check either for visibility redundancy (Algorithm 2) or for pixel redundancy (Algorithm 3).

Figure 3 illustrates non-redundant rendering on a simple scene and compares it to sampling using multiple conventional depth images and to depth peeling. Non-redundant rendering uses two images I_0 and I_1. I_0 is a conventional image that stores the first sample visible along each ray. I_0 captures the front faces of blocks A and D, as shown in the first row of Fig. 4. I_1 is rendered differentially and it captures the front faces of blocks B and C, as shown in the third row of Fig. 4. For all cases, the images are rendered with back-face culling, which correctly avoids capturing the back faces of the blocks. For conventional sampling, I_0 is the same as
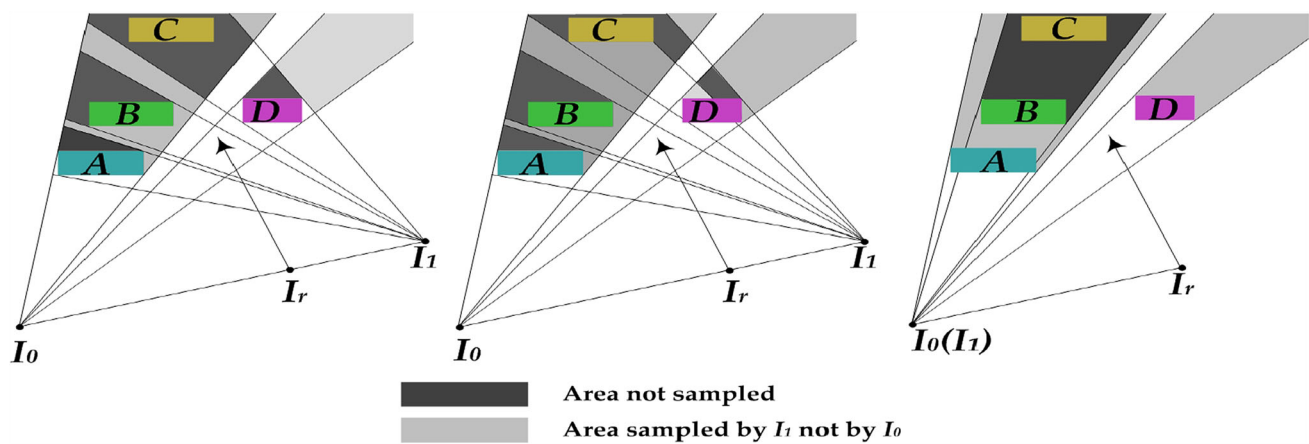
Fig. 3 2D illustration of a scene with *four blocks A–D* sampled with two images I_0 and I_1 using conventional sampling (*left*), non-redundant rendering (*middle*) and depth peeling (*right*). Non-redundant rendering captures all blocks, including block C, which is missed by the other two approaches, but which is visible in an intermediate image I_r

before, but I_1 only captures B and not C, also see second row of Fig. 4. C happens to be hidden from both I_0 and I_1, but it is visible from an intermediate viewpoint, causing disocclusion errors in I_r. Depth peeling acquires block B in the second layer (right in Fig. 3 and fourth row in Fig. 4), but block C is missed since C is doubly hidden in I_0, by both A and B, so the two layers are not enough. As shown in the right column of Fig. 4, sampling the scene with non-redundant rendering captures enough samples for a reconstruction of the intermediate image that is comparable to the ground truth image obtained by rendering redundancy definition of sample redundancy, but similar results are obtained using visibility redundancy.

The work presented here does not optimize viewpoint placement, but rather improves scene sampling for a given set of viewpoints. The views from where to sample the scene are provided as input to Algorithm 1. These views could be, for example, defined at the two endpoints of a viewpoint translation segment, at the three corners of a viewpoint translation triangle, or at the four corners of the rectangle of an area light source, in the case of soft shadow rendering, as explained in Sect. 5.

4 Reconstruction

Most applications of scene discretization need a method for reconstructing the scene from the samples captured. For example, in remote visualization, the client has to reconstruct the current frame from the samples transmitted from the server. In shadow rendering, visibility has to be evaluated by intersecting light rays with the sampled geometry.

There are two main reconstruction approaches: with explicit connectivity, i.e., the mesh approach, and without explicit connectivity, i.e., point-based approach. The mesh approach uses the connectivity information implicitly defined by the regular structure of a depth image: four neighboring samples are connected using two triangles, unless they are separated by a depth discontinuity. As GPUs have grown more powerful, rendering two triangles per depth image pixel can be done efficiently. Multiple point-based rendering approaches have been developed that bypass the need of sample connectivity. The challenge for these methods is estimating the output image footprint of the sample accurately enough to avoid holes between neighboring samples, while avoiding excessive overdraw.

In the case of non-redundant rendering, avoiding redundancy comes at a cost of a more fragmented image. A non-redundantly rendered image has more discontinuities than a conventional image as a surface has to be discontinued to avoid sampling the surface more than once. In other words, the redundant parts of a conventional image are extricated and replaced with parts of different surfaces, and each transition creates a discontinuity.

The fragmented nature of non-redundantly rendered images makes both reconstruction approaches more challenging. The mesh approach underestimates the reconstructed surface by leaving a one pixel gap in between samples separated by a discontinuity. An isolated sample, that is a sample that is not connected to any of its eight immediate neighbors, is discarded and does not contribute to the reconstructed surface. When discontinuities abound, like in the case of non-redundant rendering, this approximation error is severe. The point-based approach provides only coarse approximations of the footprint of isolated samples, a problem that is exacerbated in non-redundant rendering.

We have developed a hybrid reconstruction method that achieves a watertight reconstruction of a surface without overdraw, and that is suitable for highly fragmented
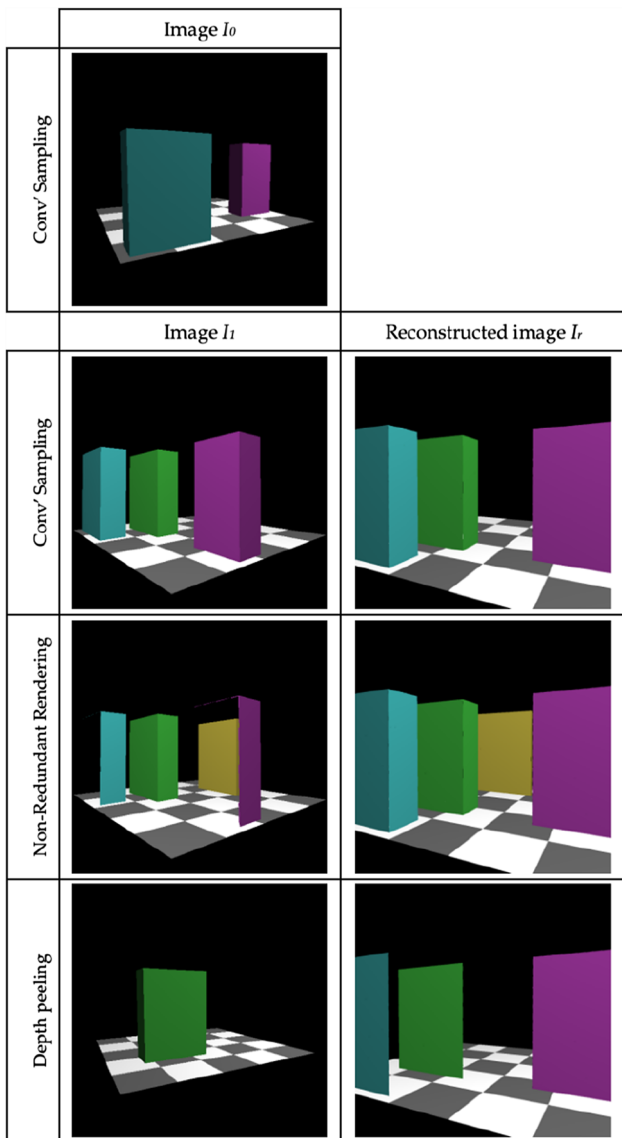
Fig. 4 Scene with four blocks from Fig. 3 sampled with two images using multiple approaches, and image reconstructed from the samples captured by the two images

Algorithm 4 HybridReconstruction

Input: non-redundant image I, output image viewpoint e.
Output: 3-D triangle mesh M that corresponds to the scene reconstruction according to I.

1: Render non-redundant image H with the same view and same resolution as I, but offset half a pixel in both directions
2: **for each** pixel p in I storing a 3D sample point s with normal n **do**
3:    **for each** corner c of p that is missing from H **do**
4:       Define ray r from e through c
5:       Define plane p through s with normal n
6:       Approximate c as intersection of r with p
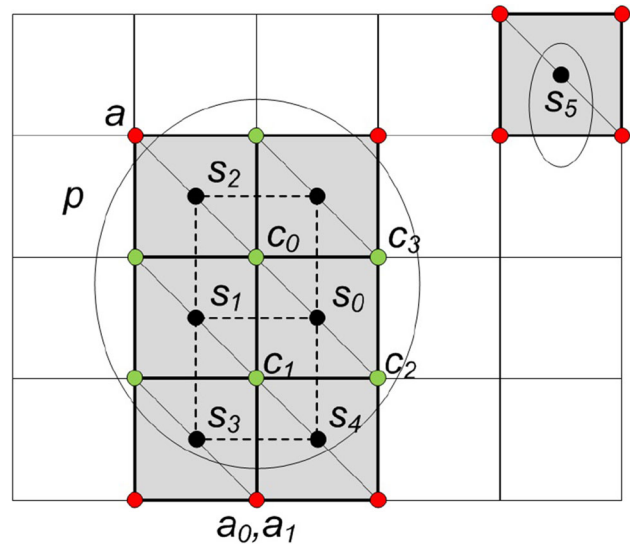7:   Generate triangles $c_0c_1c_2$, $c_2c_3c_0$ and add them to M

Fig. 5 Hybrid reconstruction algorithm

sample-based representations like the one produced by non-redundant rendering. Like the point-based approach, the hybrid reconstruction assigns a surface patch to an individual sample. Like the mesh approach, the surface patches tile perfectly by sharing vertices. Our hybrid reconstruction approach proceeds according to Algorithm 4.

The hybrid reconstruction is illustrated in Fig. 5. An image I of 5×4 resolution samples two ellipses. The point samples captured at the centers of the pixels of I are shown with black dots. The conventional mesh reconstruction, shown with dotted lines, only creates four triangles for the big ellipse, which under-approximates geometry. Much of the big ellipse is discarded, and the small ellipse does not appear in the recon-

struction at all as it is only sampled by an isolated sample which is discarded by the conventional reconstruction. The point samples captured at the centers of the pixels of the offset image H are shown in green. These samples correspond to the corners of the pixels in I. The hybrid reconstruction algorithm converts each sample in I to a quadrilateral modeled with two triangles. Sample s_0 has four valid corners in H, i.e., c_0, c_1, c_2, and c_3. A valid corner is a corner that is not separated from the center sample by a depth discontinuity. Sample s_1 uses the exact same corner samples c_0 and c_1 that are used by sample s_0, which makes the reconstruction mesh of a continuous surface watertight, and with no overdraw. Sample s_2 is missing its a corner in H, which is approximated using the normal of s_2. The corners a_0 and a_1 generated for samples s_3 and s_4 have the same image coordinates but they are different 3D points resulting from the intersection of the same corner ray with different planes as given by the different normals at s_3 and s_4. Sample s_5 is completely isolated, with no valid corner in H; however, the sample is not discarded and it contributes to the reconstruction with a quad defined by the four approximated corners.

The resulting reconstruction provides a good approximation of scene geometry: the reconstruction of a continuous surface is watertight and non-redundant, and isolated samples contribute to the reconstruction. Conventional and hybrid reconstruction have comparable cost since both methods produce two triangles per sample. The only difference is that hybrid reconstruction needs an additional rendering pass for the offset image that is used to approximate the pixel corners. Like the conventional mesh reconstruction approach, our reconstruction under-approximates geometry when a surface partially covers a pixel but it does not cover the pixel center, such as for pixel p in Fig. 5. Our reconstruction over-approximates geometry when a surface partially covers a pixel, including the pixel center, like in the case of the pixels of s_3, s_4, and s_5. The conventional mesh reconstruction method never over-approximates geometry, so our method has the advantage of error cancellation in applications such as soft shadows, as discussed in the results section.

5 Application to soft shadows

When rendering soft shadows, the challenge is to estimate visibility between the area light source and the scene points sampled by the output image. Many methods discretize the light source into points and set out to estimate visibility between all output image samples and all light samples. Adequate soft shadows require sampling area light source at high resolution (e.g., 16×16, 32×32), which translates in having to evaluate visibility along billions of light rays. Approximating the scene geometry through discretization can accelerate the visibility computation. Instead of estimating visibility using the original geometry, visibility is estimated with a smaller cost using the scene approximation provided by the discretization. Non-redundant scene discretization outperforms conventional depth image discretization, and the advantage translates to soft shadow rendering. First, the better scene approximation provided by non-redundant depth images alleviates the light leaks caused by the geometry underestimation of conventional depth images. Second, non-redundant images allow estimating visibility without the unnecessary cost of processing the same scene samples multiple times.

We use non-redundant scene discretization in the context of soft shadows with the following algorithm: The scene is discretized non-redundantly from the four corners of the area light source (Step 1). The discretization is converted to a triangle mesh approximation of scene geometry using the reconstruction algorithm presented earlier (Step 2). Then, the reconstruction triangle meshes are used to approximate shadow mapping for each of the light samples (Step 3).

6 Results and discussion

We have tested non-redundant rendering by sampling several scenes and by comparing the results to sampling with multiple conventional images and with depth peeling. We briefly describe our implementation of the sampling techniques, we define the metrics used in the comparison, we present and discuss the comparison results, and we discuss the limitations of scene sampling by non-redundant rendering.

6.1 Non-redundant rendering

6.1.1 Implementation overview

Non-redundant rendering based on pixel redundancy is implemented with a fragment shader that rejects a sample that is redundant with any of the previously rendered images. The triangle id is passed down to the fragment shader using a geometry shader. The previous images are passed in as textures, with depth and triangle id per texel. The redundancy check is performed according to Algorithm 3. The fragment shader outputs pixels that in addition to color also store depth and the triangle id to be used when rendering subsequent images.

Regarding storage, our non-redundant images do not take any additional space compared to a conventional image. We use 72 bits per sample: 24 for color, 16 for depth, and 32 for the normal at the sample. We use the normal for reconstruction as explained in Algorithm 4, but it can also be used for relighting. A 1024×1024 takes 9MB, which is a reasonable amount of storage even for today's thinnest of clients such as phones.

Non-redundant rendering based on visibility redundancy is implemented with ray tracing according to Algorithm 2. We use NVIDIA's Optix [14] ray tracer with bounding volume hierarchy (BVH) acceleration.

Depth peeling is implemented similarly to pixel redundancy non-redundant rendering, except that the same viewpoint is used for all the images.

6.1.2 Sampling quality metrics

Consider a scene S modeled with triangles that is sampled with n reference images I_i with views PPC$_i$. We quantify sampling quality with three view-independent and one view-dependent metric. Either pixel or visibility redundancy can be used with each metric.
View-independent metrics

1. The number of *redundant samples* is defined as the number of samples in a reference image I_i that are redundant with one of the previous reference images I_j, $j < i$. By construction, there are no redundant samples for

non-redundant rendering or depth peeling. This metric measures the redundancy of conventional sampling.

2. The number of *disoccluded samples* is defined as the number of samples in a reference image that are not captured by conventional reference images. For an image that is rendered differentially from view PPC_i, a disoccluded sample is a sample that is not captured by a conventional image rendered from PPC_i. For an image that is rendered by depth peeling, a disoccluded sample is a sample that is not captured by any conventional image I_i (i from 0 to $n - 1$). For a conventional image, the number of disoccluded samples is 0. The number of disoccluded samples measures the occlusion avoidance capability of non-redundant rendering and of depth peeling.

3. The number of *useful samples* is defined as the number of disoccluded samples in a reference image that is needed in at least one output frame with an intermediate view. Such an output frame is rendered with a view PPC that interpolates in between the reference views PPC_i. (i from 0 to $n - 1$). For example, when $n = 2$, the output frames have viewpoint on the segment defined by the viewpoints of PPC_0 and PPC_1. When $n = 3$, the output frames have viewpoint on the triangle defined by the viewpoints of PPC_0, PPC_1, and PPC_2. The metric is implemented by rendering a large number of output frames (i.e., 1000) from intermediate views and by checking which reference image samples are needed in each output frame. A sample is needed in an output frame if the sample is redundant with that frame. This metric distinguishes between two types of additional samples contributed by non-redundant rendering and by depth peeling. The useful additional samples are the ones that help approximate visibility from the region defined by the n reference viewpoints. A useful sample becomes visible from an intermediate viewpoint. In other words, the sample is beneficial to the reconstruction of an intermediate image. A sample that is not useful is a sample that is never visible, from any intermediate viewpoint, and that unnecessarily reduces the occlusion culling efficiency of the set of images. *View-dependent metric*

4. The number of *output frame samples that are available* in at least one of the reference images. Whereas metric (3) reports the number of samples that are useful at least from one intermediate view, this view-dependent metric (4) reports how many samples of a given output frame can be found in the reference images. This metric is evaluated by rendering a ground truth frame from geometry and then by checking the redundancy of each of the ground truth frame samples against the set of reference images. If a sample is redundant with a reference image, then it means that the sample is available in that reference image. A sample that is not available in any of the reference images creates a disocclusion error. This

Table 1 View-independent sampling metrics

Model	Image 0 (%)	Image 1 (%)	Image 2 (%)
Redundant samples for conventional sampling			
BirdNest	0	70	80
Grass	0	54	69
Urban	0	82	84
Tree	0	23	35
Disoccluded and (useful) samples for non-redundant rendering			
BirdNest	0 (0)	62 (89)	76 (86)
Grass	0 (0)	56 (85)	75 (80)
Urban	0 (0)	73 (40)	80 (23)
Tree	0 (0)	84 (99)	88 (98)
Disoccluded and (useful) samples for depth peeling			
BirdNest	10 (99)	46 (2)	52 (71)
Grass	15 (99)	51 (4)	71 (79)
Urban	1 (100)	78 (7)	88 (8)
Tree	44 (99)	63 (98)	72 (97)

view dependent metric checks whether there is an output frame for which the reference images miss many samples leading to substantial disocclusion errors.

6.1.3 Quality

We used the four metrics above to quantify and compare non-redundant rendering to conventional multiple image sampling and to depth peeling.

Table 1 gives the figures for the three view independent metrics for the *Grass* (*56K triangles*), *Urban* (*50K triangles*), *Bird Nest* (*67K triangles*) *and Tree* (*113K triangles*) models. The useful samples are given relative to the disoccluded samples. The models are sampled using three images. Conventional sampling has a considerable number of redundant samples in *Image* 1 and 2. Non-redundant rendering and depth peeling do not have any redundant samples. Depth peeling brings disoccluded samples in *Image* 0 because *Image* 0 is rendered from the center of the triangle defined by the three viewpoints used in conventional and non-redundant sampling. The center viewpoint captures samples not visible from any of the three corner viewpoints.

The visualization of the *disoccluded* and *useful* samples for *Bird Nest*, *Grass*, *Urban*, and *Tree* in Fig. 6 Non-redundant rendering disoccludes more samples and more of the disoccluded samples are useful compared to depth peeling. The only exception is for the *Urban* model where depth peeling disoccludes slightly more samples than non-redundant rendering; however, only a small percentage of these samples are useful, so even in this case non-redundant rendering captures more useful samples than depth peeling. Tables 1 and 2 rely on pixel redundancy. As image
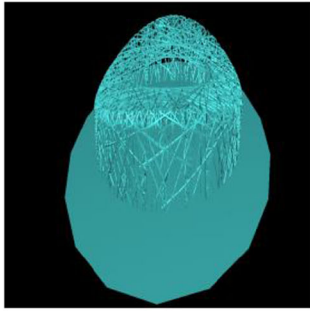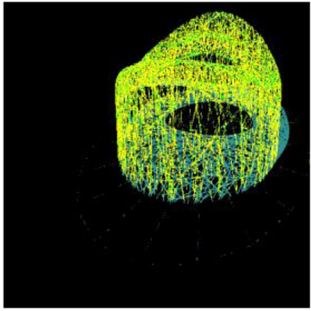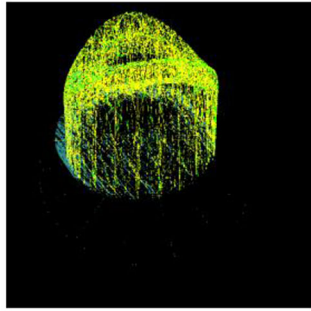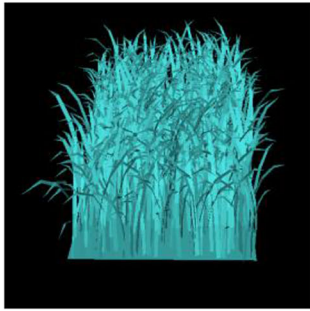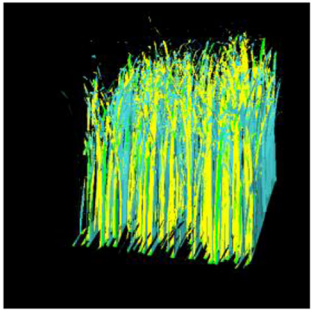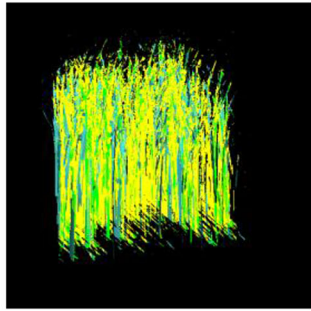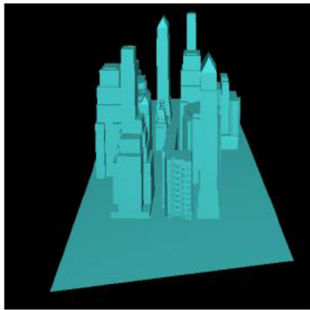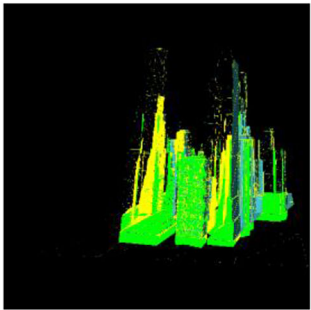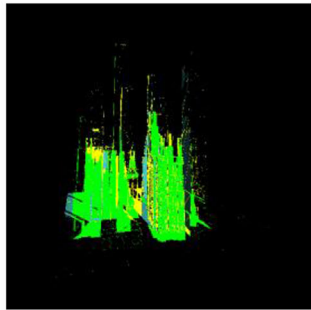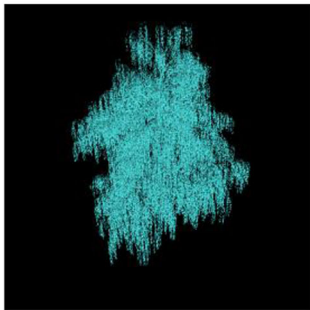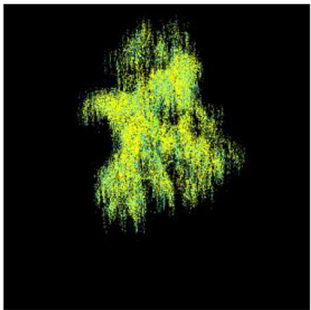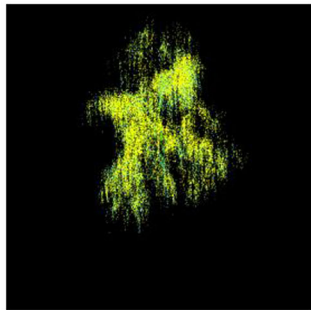
Model	Image 0	Image 1	Image 2
Bird Nest	0%, 0%	62% ,89%	76%,86%
Grass	0%, 0%	56%, 89%	75%, 79%
Urban	0%, 0%	73%,40%	80%, 24%
Tree	0%, 0%	84%, 99%	88%, 98%

Fig. 6 Illustration of disoccluded samples (*green + yellow*), useful (yellow)

resolution2 increases, pixel redundancy converges to visibility redundancy. Figure 7 shows the convergence of the number of redundant, disoccluded, and useful disoccluded samples computed using pixel redundancy to the numbers computed using visibility redundancy as image resolution increases.

Table 2 View-dependent sampling metric

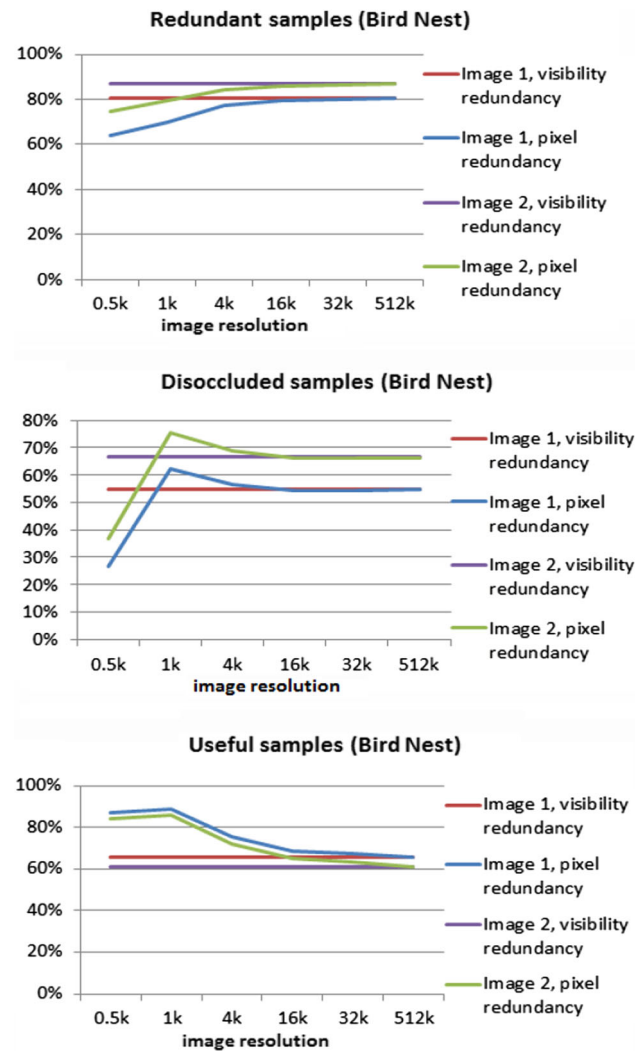| Model	Conventional sampling		Non-redundant rendering		Depth peeling	
	Avg (%)	Min (%)	Avg (%)	Min (%)	Avg (%)	Min (%)
BirdNest	90	89	93	92	92	89
Grass	86	81	96	91	93	88
Urban	97	96	98	97	96	91
Tree	55	53	59	57	55	48

Fig. 7 Dependency of number of redundant, disoccluded, and useful samples on image resolution, for both the pixel and the visibility redundant metrics. The metrics converge as resolution increases (4k means 4096 × 4096)

6.2 Application to soft shadows

We have tested non-redundant scene discretization in the context of soft shadow rendering for our test scenes. Figure 8 compares our results (column 2) to shadow map-ping

(column 1), to conventional depth image scene discretization (column 3), and to depth peeling scene discretization (column 4). In all cases, the depth image resolution is $1,024 \times 1,024$, the output image resolution is 512×512, the area light source is sampled at a 16×16 resolution, and the area light source diagonal equals the diagonal of the bounding box of the scene. The shadow mapping images were rendered by rendering a shadow map from the original scene geometry, for each of the light samples. For the non-redundant and the conventional depth image approaches, the depth images are rendered from the four corners of the area light source. The depth images for depth peeling are rendered from the center of the area light source. Since the three scene discretization methods investigated are based on an approximation of the shadow maps of each of the light samples, we use shadow mapping as ground truth for estimating the errors brought by the discretization methods. The figures and tables report two quantitative measures of the soft shadow approximation error. The visibility error is defined as the per pixel average of the number of visibility rays that are resolved incorrectly. Since the light is sampled at 16×16 resolution, the visibility error ranges from 0 to 255. The intensity error is defined as the average per-pixel shadow level error, with a range from 0 to 255. The visibility error is a sum of the absolute values of individual visibility errors, whereas the intensity error is an algebraic sum of visibility errors. An incorrect in-light determination at a pixel cancels out with an incorrect in-shadow determination made at the same pixel. Therefore, the visibility error is always larger than the intensity error.

Non-redundant scene discretization has lower errors than conventional depth image or depth peeling scene discretizations. The *Tree* scene is the most challenging scene for all methods, as completely removing the complex occlusions requires more than four depth images. Our method is more robust than the previous scene discretization approaches, as it performs well for all scenes. Depth peeling works poorly for scenes such as *Urban*, and conventional depth images work poorly for scenes such as *BirdNest*, and *Grass*.

Table 3 shows the dependency of the intensity (i) and visibility (v) errors on the depth image resolution, for all three scene discretization approaches. For brevity, only the figures for the *Urban* and *Grass* scenes are given. As expected, the error decreases as resolution increases. Our method maintains its advantage at all resolutions. Table 4 shows the dependency of the visibility and the intensity errors on the size of the area light source. The area light source size is given in fractions and multiples of the scene bounding box diagonal. As expected, the errors go up as the light source size increases, since the four depth images have to cover an increasing range of viewpoints. As expected, the errors are always larger for conventional depth images than for non-redundant rendering since the set of samples captured by our method is a superset of the samples captured by conventional
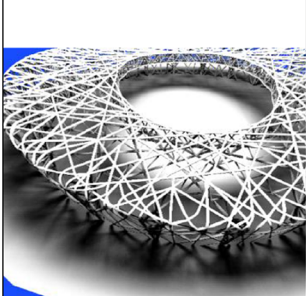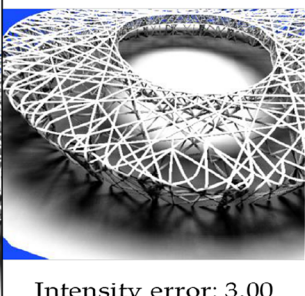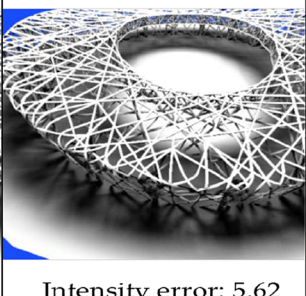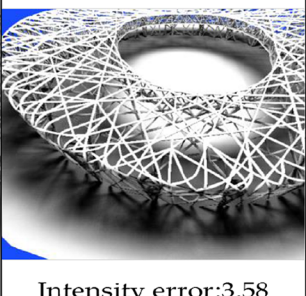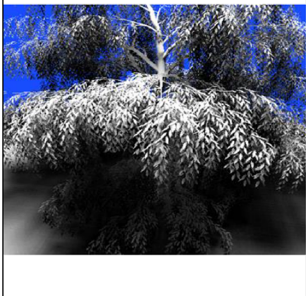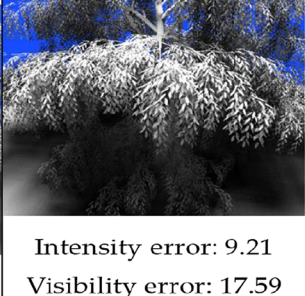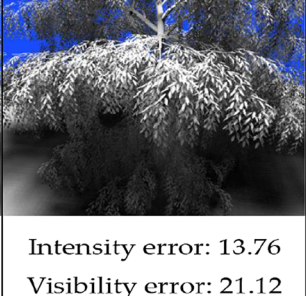
Fig. 8 Soft shadows rendered using the original geometry and using the three scene discretization approaches. Our non-redundant discretization approach produces the most accurate shadows (also see Fig. 1)

Table 3 Soft shadow error dependency on discretization resolution

Depth image resolution	Non-redundant rendering	Conv. sampling	Depth peeling
Grass			
512	i: 3.20	i: 7.98	i: 4.57
	v: 7.21	v: 9.62	v: 8.34
1024	i: 1.71	i: 5.15	i: 2.55
	v: 3.70	v: 6.12	v: 4.41
1536	i: 1.26	i: 4.24	i: 1.84
	v: 2.55	v: 4.94	v: 3.02
Urban			
512	i: 3.54	i: 7.40	i: 15.85
	v: 4.74	v: 7.80	v: 18.52
1024	i: 1.60	i: 3.46	i: 11.36
	v: 2.07	v: 3.64	v: 12.62
1536	i: 1.10	i: 2.28	i: 9.97
	v: 1.36	v: 2.39	v: 10.77

depth images. Depth peeling has occasionally a slight advantage over our method for small area light sources, but this comes at the cost of unpredictable performance, with errors that can be quite large. For example, in the case of the *Urban* scene, the depth peeling intensity error is 11.4 for a light size of 1, compared to 1.60 for non-redundant rendering.

Table 5 shows the times for each of the three main steps of the soft shadow rendering algorithm given in Algorithm 5, for each of three scene discretization approaches. The times were measured on a workstation with a 3.4GHz Intel(R) Core(TM) i7-2600 CPU, with 4GB of memory, and with an NVIDIA GeForce GTX 570 graphics card. The three approaches have similar performance. Non-redundant rendering of the depth images is only marginally slower than rendering conventional depth images, and comparable to depth peeling. The slowest step is step 3 which renders a shadow map for each light sample from the reconstruction meshes.

The time to compute the approximation of the scene geometry is the sum of the times for steps 1 and 2, which ranges from 75ms for the *Tree* and 116ms for *Urban*. This shows that the approximation can be computed at interactive rates, supporting dynamic scenes where geometry and lights move, deform, and change size. Out of the three steps, it is only Step 1 whose performance depends on the complexity of the original scene. Step 1, which performs non-redundant sampling with a small modification of the conventional interactive graphics pipeline, is very efficient (i.e., a few milliseconds) and, therefore, our method scales well with scene complexity.

For scene discretization to have an advantage over rendering soft shadows using conventional shadow mapping or ray tracing, the meshes that result from scene reconstruction have to be less complex than the original scene geometry. In

Table 4 Soft shadow error dependency on light size

Area light size	Non-redundant rendering	Conv. sampling	Depth peeling
Grass			
1/4	i: 1.53	i: 3.57	i: 2.53
	v: 3.057	v: 4.16	v: 4.22
1/2	i: 1.62	i: 4.24	i: 2.54
	v: 3.40	v: 4.88	v: 4.31
1	i: 1.71	i: 5.15	i: 2.55
	v: 3.70	v: 6.12	v: 4.41
2	i: 1.88	i: 5.55	i: 2.79
	v: 3.97	v: 6.92	v: 4.68
4	i: 2.83	i: 5.95	i: 3.59
	v: 4.98	v: 7.86	v: 5.40
Urban			
1/4	i: 2.38	i: 4.94	i: 3.95
	v: 3.39	v: 5.13	v: 5.89
1/2	i: 2.06	i: 3.81	i: 5.98
	v: 2.86	v: 4.01	v: 7.50
1	i: 1.60	i: 3.46	i: 11.36
	v: 2.07	v: 3.64	v: 12.62
2	i: 1.07	i: 2.67	i: 19.91
	v: 1.38	v: 3.01	v: 21.27
4	i: 1.95	i: 6.77	i: 29.64
	v: 2.30	v: 7.35	v: 31.11

Table 5 Times for soft shadow rendering steps

Scene	Non redundant rendering	Conv. sampling	Depth peeling
Step 1: rendering depth image (ms)			
B'nest	3.9	2.2	4.2
Grass	4.8	2.2	5.1
Urban	4.4	2.1	3.6
Tree	7.2	5.2	8.6
Step 2: reconstruction (ms)			
B'nest	55	90	44
Grass	88	117	82
Urban	112	131	98
Tree	68	94	57
Step 3: rendering soft shadows (ms)			
B'nest	710	771	490
Grass	735	926	642
Urban	802	976	541
Tree	807	954	628

other words, the scene geometry has to be complex enough for the reconstruction meshes to bring a simplification of the scene geometry. Figure 9 shows the frame times for the

Algorithm 5 SoftShadowRendering

Input: rectangular area light source L, scene S, output image I w/o shadows.
Output: fractional visibility of L for each sample of I.
1: Render non-redundant depth images $NRDI$ of S from each of the four corners of L
2: **for each** $NRDI_i$ **do**
3:    compute the scene reconstruction mesh M_i
4: **for each** light sample s **do**
5:    Initialize approximate shadow map SM
6:    **for each** M_i **do**
7:        Render M_i from viewpoint s onto SM
8:    **for each** M_i **do**
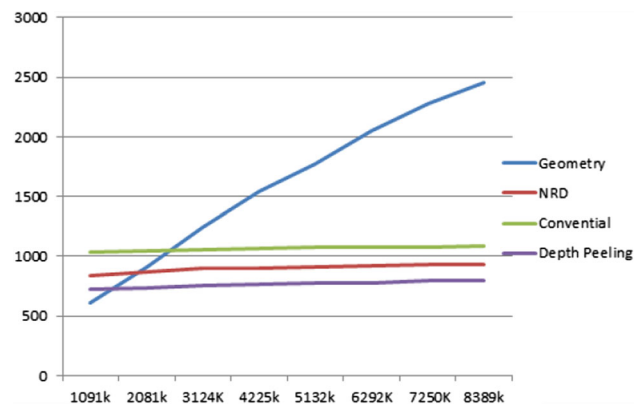9:   Estimate visibility from s to o using SM

Fig. 9 Soft shadow frame times as a function of scene complexity

three scene discretization methods compared to rendering the shadow maps for the light samples directly from the original geometry. The Grass scene with varying degrees of complexity is used. The scene discretization approaches have similar performance and start outperforming the shadow mapping approach beyond the 250,000 triangle scene complexity level.

6.3 Limitations

One limitation noted above is the slight rendering performance penalty brought by having to check for sample redundancy. However, non-redundant rendering performance is always comparable to depth peeling and it has the benefit of additional viewpoints which translates into better sampling quality (Sect. 6.1.3). Like for depth peeling, non-redundant rendered images have lower sample coherence than a conventional image, as avoiding sample redundancy implies image fragmentation. This lower coherence complicates reconstruction from the samples of the non-redundant rendered images, since sample connectivity is more difficult to infer. Another repercussion of this lower coherence is that non-redundant rendered images do not compress as well as conventional images.

Finally, even though non-redundant rendering brings more of the samples needed from a region, there is still no guarantee that all samples needed are captured. Conversely, there is no guarantee that all the new samples brought in through non-redundant sampling are actually needed by the application. Our experiments show that most of the new samples are visible from at least one nearby viewpoint, hence most of the new samples are useful. Compared to conventional images, non-redundant rendered images are more powerful aggressive visibility solutions as they replace samples that are known not to be useful with samples that are likely to be useful. This helps achieve adequate scene sampling with fewer images, and better scene sampling using the same number of images. The problem of deciding how many images are needed and from what viewpoints such that all scene samples visible from a view region are captured is NP complete. Therefore, applications involving dynamic scenes, where the discretization has to be computed on the fly, will continue to rely on approximate, greedy solutions. Non-redundant rendering improves the quality of the set of samples that can be acquired quickly.

Like any scene discretization method, non-redundant rendering pays off for scenes sufficiently complex such that the number of scene triangles is substantially larger than the number of samples resulting from discretization. In the work presented here, the original scene geometry is always completely replaced by the geometry reconstructed from the non-redundantly sampled depth images. Future work could consider resorting to a hybrid approach that leverages the original geometry where appropriate, e.g., simple scenes, nearby geometry, and reserves the use of the reconstructed geometry for complex/distant parts of the scene.

In the case of soft shadow rendering, sampling the four corners of the light is a heuristic. Even though non-redundant images have greater sampling capability, there is no guarantee that sufficient samples are captured. For complex scenes such as Tree, additional images would help. One could use a trial and error approach for deciding how many images are needed for a scene–add images until the error dips below an application selected threshold.

The average soft-shadow rendering errors are small but the maximum error can be large. For example, for Figs. 1 and 8, the maximum per pixel intensity error for our approach is 106, 112, 48, and 126 for the four scenes, respectively. This is due to the fact that visibility does not vary smoothly in complex scenes, and for occasional pixels a large number of visibility rays are estimated incorrectly. The maximum errors are smaller for non-redundant rendering compared to conventional depth images (errors of 179, 181, 74, and 126), and to depth peeling (errors of 137, 199, 125, and 148).

7 Conclusions and future work

We have presented non-redundant rendering, a simple method for improving the scene sampling efficiency of conventional images. The main idea is to detect and avoid sample redundancy as the second and subsequent images are rendered. We have shown that non-redundant rendering compares favorably to conventional sampling and to depth peeling according to several view-independent and view-dependent metrics. Compared to depth peeling, non-redundant rendering preserves the advantage of avoiding redundant samples, but non-redundant rendering samples the space of rays more uniformly, relying on more than a single viewpoint, which translates in improved sampling quality. We have then presented a scene reconstruction method that can handle the higher degree of fragmentation characteristic to non-redundant images. Finally, we have applied our non-redundant rendering and scene reconstruction methods to the context of soft shadow rendering, where we have demonstrated quality advantages over prior-art scene discretizations such as conventional images and depth peeling.

Performing depth peeling from several viewpoints will improve sampling over depth peeling for a single viewpoint. However, the approach would have two important disadvantages. First, depth peeling from multiple viewpoints requires an increased number of images. For example, three viewpoints, each with three depth peeling layers, result in nine images. For our non-redundant rendering sampling approach, we typically use three images total. Second, whereas there are no redundant samples between the depth peeling layers of one viewpoint, there will be redundancy between the layers of the different viewpoints. In particular, there will be significant redundancy between the first layers of each viewpoint, as the first layers are conventional depth images. In conclusion, our approach of non-redundant sampling achieves simultaneously both a change of viewpoint and avoidance of redundancy, which leads to more efficient sampling.

Future work directions include considering graphics architecture changes to provide better support to non-redundant rendering, as well as incorporating non-redundant rendering in additional applications such as visibility computation, remote visualization, or reflection rendering.

References

1. Agarwala, A., Agrawala, M., Cohen, M., Salesin, D., Szeliski, R.: Photographing long scenes with multi-viewpoint panoramas. ACM Trans. Graph. (TOG) **25**(3), 853–861 (2006)
2. Bavoil, L., Callahan, S.P., Lefohn, A., Comba, J.L.D., Silva, C.T.: Multi-fragment effects on the gpu using the k-buffer. In: Proceedings of the 2007 Symposium on Interactive 3D Graphics and Games, pp. 97–104. ACM, New York (2007)
3. Bavoil, L., Myers, K.: Order independent transparency with dual depth peeling. In: NVIDIA OpenGL SDK, pp. 1–12 (2008)
4. Chang, C.F., Bishop, G., Lastra, A.: Ldi tree: a hierarchical representation for image-based rendering. In: Proceedings of the 26th Annual Conference on Computer Graphics and Interactive Techniques, pp. 291–298. ACM Press/Addison-Wesley Publishing Co., New York (1999)
5. Eisemann, E., Décoret, X.: On exact error bounds for view-dependent simplification. In: Computer Graphics Forum, vol. 26, pp. 202–213. Wiley Online Library, New York (2007)
6. Everitt, C.: Interactive order-independent transparency. In: White Paper nVIDIA **2**(6), 7 (2001)
7. Liu, B., Wei, L.Y., Xu, Y.Q., Wu, E.: Multi-layer depth peeling via fragment sort. In: In: 11th IEEE International Conference on Computer-aided design and computer graphics, 2009. CAD/Graphics' 09, pp. 452–456. IEEE, New York (2009)
8. Maciel, P.W.C., Shirley, P.: Visual navigation of large environments using textured clusters. In: Proceedings of the 1995 Symposium on Interactive 3D Graphics, pp. 95–ff. ACM, New York (1995)
9. Mark, W.R., McMillan, L., Bishop, G.: Post-rendering 3d warping. In: Proceedings of the 1997 Symposium on Interactive 3D Graphics, pp. 7–ff. ACM, New York (1997)
10. Matthias, N., Henry, S., Marc, S.: Fast indirect illumination using layered depth images. Vis. Comput. (TVCJ) **26**(6), 679–686 (2010)
11. Max, N., Ohsaki, K.: Rendering trees from precomputed z-buffer views. In: Rendering Techniques, vol. 95, pp. 74–81. Springer, Berlin (1995)
12. McMillan, L., Bishop, G.: Plenoptic modeling: an image-based rendering system. In: Proceedings of the 22nd Annual Conference on Computer Graphics and Interactive Techniques, pp. 39–46. ACM, New York (1995)
13. Mei, C., Popescu, V., Sacks, E.: The occlusion camera. In: Computer Graphics Forum, vol. 24, pp. 335–342. Wiley Online Library, New York (2005)
14. NVIDIA ®. Optix™ ray tracing engine. https://developer.nvidia.com/optix
15. Rademacher, P., Bishop, G.: Multiple-center-of-projection images. In: Proceedings of the 25th Annual Conference on Computer Graphics and Interactive Techniques, pp. 199–206. ACM, New York (1998)
16. Roman, A., Garg, G., Levoy, M.: Interactive design of multi-perspective images for visualizing urban landscapes. In: Proceedings of the Conference on Visualization'04, pp. 537–544. IEEE Computer Society, New York (2004)
17. Román, A., Lensch, H.P.A.: Automatic multiperspective images. In: Rendering Techniques **2**(2006), 161–171 (2006)
18. Shade, J., Gortler, S., He, L.W., Szeliski, R.: Layered depth images. In: Proceedings of the 25th Annual Conference on Computer Graphics and Interactive Techniques, pp. 231–242. ACM, New York (1998)
19. Nguyen, K.T., Hanyoung, J., JungHyun, H.: Layered occlusion map for soft shadow generation. Vis. Comput. (TVCJ) **26**(12), 1497–1512 (2010)
20. Wilson, A., Manocha, D.: Simplifying complex environments using incremental textured depth meshes. In: ACM Transactions on Graphics (TOG), vol. **22**, pp. 678–688. ACM, New York (2003)
21. Wonka, P., Wimmer, M., Zhou, K., Maierhofer, S., Hesina, G., Reshetov, A.: Guided visibility sampling. ACM Trans. Graph. (TOG) **25**(3), 494–502 (2006)
22. Yu, J., McMillan, L.: General linear cameras. In: Computer Vision-ECCV 2004, pp. 14–27. Springer, Berlin (2004)

Naiwen Xie received his B.E. degree in computer science from the Huazhong University of Science and Technology in 2011. He is currently working toward his Ph.D. degree in the State Key Laboratory of Virtual Reality Technology and Systems at Beihang University. His research interests include global illumination, image-based rendering, and image processing.

Lili Wang received the Ph.D. degree from the Beihang University, Beijing, China. She is an associate professor with the School of Computer Science and Engineering of Beihang University, and a researcher with the State Key Laboratory of Virtual Reality Technology and Systems. Her interests include real-time rendering, realistic rendering, global illumination, soft shadow and texture synthesis.

Voicu Popescu received a B.S. degree in computer science from the Technical University of Cluj-Napoca, Romania in 1995, and a Ph.D. degree in computer science from the University of North Carolina at Chapel Hill, USA in 2001. He is an associate professor with the Computer Science Department of Purdue University. His research interests lie in the areas of computer graphics, computer vision, and visualization. His current projects include camera model design, visibility, augmented reality for surgery telementoring, and the use of computer graphics to advance education.