This paper presents the model and rendering algorithm of Potentially Visible Hidden Volumes (PVHVs) for multi-view image warping. PVHVs are 3D volumes that are occluded at a known source view, but potentially visible at novel views. Given a bound of novel views, we define PVHVs using the edges of foreground fragments from the known view and the bound of novel views. PVHVs can be used to batch-test the visibilities of source fragments without iterating individual novel views in multi-fragment rendering, and thereby, cull redundant fragments prior to warping. We realize the model of PVHVs in Depth Peeling (DP). Our Effective Depth Peeling (EDP) can reduce the number of completely hidden fragments, capture important fragments early, and reduce warping cost. We demonstrate the benefit of our PVHVs and EDP in terms of memory, quality, and performance in multi-view warping.

1 INTRODUCTION

In novel-view warping, multi-fragment rendering can be thought of as a primary solution to handle disocclusions using hidden surfaces at the known view. Multi-fragment rendering techniques, including A-buffer [Carpenter 1984], k-buffer [Bavoil et al. 2007], and Depth Peeling (DP) [Everitt 2001; Mammen 1989], capture many fragments for transparency or global illumination. However, for novel-view warping, many of the fragments are redundant, which are invisible from any views and do not contribute to the final outcome.

Our key insight in this work is early test of the visibilities of fragments to be warped. This approach can cull redundant fragments in multi-fragment rendering, and better capture important fragments, using fewer layers in DP. However, this poses a challenge. Fragments are captured for the known view, but their visibilities require to be tested against novel views. Hence, disocclusions, in general, are revealed after warping, where the warping even requires to be iterated for multiple views. Although if tests are possible, the novel views can be too many to test, or unavailable in advance.

We tackle this challenge by modeling Potentially Visible Hidden Volumes (PVHVs), which are hidden at the known source view but visible at novel views. Our approach is similar to the shadow volumes [Crow 1977; Eisemann et al. 2011] in terms of the use of 3D volumes to determine visibilities. Our approach does not require individual novel views but just a coarse yet conservative bound of potential view displacements. Given the bound, we geometrically analyze visibilities and find PVHVs. Thereby, the PVHVs can cull completely occluded fragments early (prior to warping). In this way, an effective multi-fragment rendering problem can be isolated without knowing the number of views and their precise displacements.

This paper presents the definition, modeling, and rendering of PVHVs for effective multi-fragment rendering for multi-view warping. The key idea to estimate PVHVs is based on the observation that the visibilities of fragments can be determined using the edges of foreground fragments from the known view and the bound of novel view positions. To prove the utility of PVH, we realize it in DP, which renders hidden layers in order. Our Effective DP (EDP), using PVHVs, can reduce completely hidden fragments, capture
important fragments early, and reduce warping cost (see Fig. 1 for example).

Our PVHVs can improve multi-fragment rendering in terms of memory, performance, and quality for multi-view warping applications, such as Depth-of-Field (DOF) rendering. First, sparser fragments generated with PVHVs can save memory. In case of EDP, we can achieve this by tightly packing fragments. Similarly, other multi-fragment rendering techniques (e.g., A-buffer or k-buffer) can also cull redundant fragments; in this case, PVHV-based culling can be a post-processing orthogonal to fragment capture. Albeit in different contexts, similar ideas have been explored in remote rendering, which bypasses a foreground layer for warping [Reinert et al. 2016] or culls hidden triangles for potential viewpoints [Hladky et al. 2019]. Second, our EDP can use on-the-fly fragment culling. The per-layer cost of EDP is costlier than the regular DP, but important fragments are captured in earlier layers, which are often extracted in a deeper layer by the regular DP. Hence, we can use fewer layers for similar-quality warping, speeding up peeling. Lastly, warping cost can be reduced given similar quality. For instance, forward warping can use fewer source fragments and bypass empty areas.

In this work, we realize PVHVs for EDP, but can motivate further studies in other multi-fragment rendering, remote rendering, and video-based warping. Precisely, our contributions in this work can be summarized as follows:

- definition and modeling of potentially visible hidden volumes for multi-fragment rendering for multi-view warping;
- an effective depth peeling algorithm that realizes PVHV for on-the-fly fragment culling.

2 RELATED WORK

2.1 Depth Peeling and Its Applications

Extracting multiple fragments at the same screen-space location has unbounded memory footprint (e.g., A-Buffer [Carpenter 1984; Yang et al. 2010]). In constrained scenarios requiring bounded memory footprint, we limit the number of fragments per screen-space location (e.g., k-buffer [Bavoil et al. 2007] and DP). DP sorts fragments by multi-pass iteration, but has been popular for its simplicity and efficiency for scenes of low depth complexity. For a complete survey of multi-fragment rendering, we refer the reader to the recent review article [Vasilakis et al. 2020].

The standard DP [Everitt 2001; Mammen 1989] peels one layer per each geometry pass, and thus, its cost linearly scales with the number of geometry passes. Many of later efforts have been dedicated to the reduction of the geometry passes. Dual depth peeling [Bavoil and Myers 2008] applies front-to-back and back-to-front peelings simultaneously, and reduces the geometry passes down to a half. Bucket DP [Liu et al. 2009] sorts fragments into depth buckets allocated for sub-intervals in a single pass; the additional pre-analysis pass better distributes the sub-intervals. Deep G-buffers [Mara et al. 2016] generate two layers in a single pass by predicting foreground depths in novel views. Our PVHVs and EDP do not reduce the number of geometry passes, but can implicitly reduce the number of layers by capturing important fragments early in DP iteration.

The GPU implementation of DP was devised to handle order independent transparency [Everitt 2001], but has been later extended to handle ray tracing [Bernardon et al. 2006; Franke et al. 2018; Lee et al. 2010; Mara et al. 2016; McGuire and Mara 2014; Nagy and Klein 2003], shadow maps [Bavoil et al. 2006, 2008; Lee et al. 2018], global illumination [Bavoil and Sainz 2009; Hachisuka 2005; Mara et al. 2016; McGuire and Mara 2014], and defocus blur [Franke et al. 2018; Lee et al. 2010]. Many of these applications are also common in other multi-fragment rendering. We again refer the reader to the recent survey [Vasilakis et al. 2020] for a complete list of applications.

Our work manifests itself for multi-view warping, because PVHV analysis adds additional (but lower than single-layer) overhead to conventional DP. We demonstrate our EDP for DOF rendering.

2.2 Fragment Visibility Culling

Our PVHV solution is a sort of fragment-space visibility culling, and is similar in spirit to Potentially Visible Set (PVS) for offline geometry culling [Cohen-Or et al. 2003; Teller and Séquin 1991] or online triangle culling in remote streaming [Hladky et al. 2019]. Hladky et al. cull completely hidden triangles, while we cull completely hidden fragments. Their algorithm also operates on fragments, but identifies visible triangles per fragment. The next view is rendered with much lighter geometry passes owing to the PVSs. In contrast, our captures potentially visible fragments to use them for image-space warping. Hence, ours requires fine-grained visibility tests in much higher accuracy, and can preclude more redundancy by the fragment-space culling.

Multi-fragment rendering has used per-pixel visibility culling to capture the best fragments. Occlusion culling quickly discards already processed objects for DP [Vasilakis and Fudos 2012] or fragments (occupancy maps for k-buffer [Vasilakis and Papaioannou 2015]). Dynamic k+-buffer has proposed finding the optimal k (a pre-assigned constant number of layers) using the depth histogram analysis [Vasilakis et al. 2015]. Variable k-buffer has improved this dynamic-k scheme to the per-pixel level [Vasilakis et al. 2017]. While efficient for single-view culling, these approaches do not account for visibilities against novel views.

For novel-view warping (in particular for DP), scene- or view-dependent analysis can control the peeling thresholds for fragment culling. It can be empirically chosen considering scene characteristics [Mara et al. 2016]. Back-projecting a pixel as an occluder, in defocus blur [Lee et al. 2010] and depth warping [Lee et al. 2018], can find umbrae, and deeper leaks can be made to the next peeling thresholds. These single-fragment culling techniques are simple and efficient. However, there is still a significant room in further culling of surfaces hidden by larger-than-fragment areas, which is addressed by our edge-based geometric analysis.

2.3 Multi-Fragment Depth-of-Field Rendering

DOF rendering is one of the common applications of multi-fragment rendering. The majority of the real-time DOF rendering techniques use single-view filtering, and inherently suffer from visibility problems (e.g., partial occlusion and intensity leakage) due to the lack of hidden fragments. Such problems have been addressed by multi-fragment rendering. Single-pass discrete layer decomposition [Lee et al. 2009] and DP [Lee et al. 2010] have obtained physically-faithful image quality by including hidden fragments in the image-space ray tracing.
tracing. The efficacy of the multi-fragment rendering has been also proved in the filtering (gathering) and scattering approaches. The multi-layer filtering [Selgrad et al. 2015] based on the A-Buffer [Yang et al. 2010] and multi-layer splatting [Franke et al. 2018] have attained high-quality DOF effects.

3 POTENTIALLY VISIBLE HIDDEN VOLUME (PVHV)

In this section, we define the model of PVHV, and explain the types of PVHVs with respect to the shape of novel views.

3.1 Definition of PVHV

Let $W$ be the entire 3D volume of the scene. For a particular known view $s$, $W$ can be decomposed into mutually exclusive visible volume $V_s$ and hidden occlusion volume $O_s$, where we can see only objects in $V_s$. Given a novel view $n$, we define a PVHV $H_s(n) := O_s \cap V_n$ (Fig. 2). By definition, objects in $H_s(n)$ are unavailable in the foreground rendered for $s$, but require to be captured for $n$. Redundancy is found at $O_s - H_s(n)$, where objects and their fragments are invisible from both $s$ and $n$. Our goal is to preclude and cull $O_s - H_s(n)$ in multi-fragment rendering.

As shown in Fig. 2(c), our important observation is that PVHVs are formed by the edges from the source view $s$ and novel view $n$ through foreground occluders. This motivates our edge-based geometric analysis, which distinguishes ours from the classical occlusion culling. While the occlusion culling (and PVS) typically uses the shapes or bounding volumes of occluders, we instead utilize edge-based culling. While the occlusion culling (and PVS) typically uses edge-based culling, which distinguishes ours from the classical occlusion culling.

3.2 Types of PVHV

The shape of a PVHV, obviously, varies by the distribution of novel views. In general, novel views can be classified into linear, point, and areal views, depending on applications. For example, point views can be defined in motion-based temporal warping or spatial projection (e.g., stereoscopy). Linear views are defined from motion blur. Areal views can be defined from light field [Levoy and Hanrahan 1996; Yu et al. 2010] (rectangular), DOF rendering (circular), and soft shadow mapping. We describe how different view types define PVHVs.

3.2.1 Linear Novel Views. The simplest shape of a PVHV is defined by linear views, which reduces down to a 2D triangle (the green region in Fig. 3(a)). Here, let $n$ be the farthest view in a set of novel views $N$ that span from $s$ by a distance $E$; here, $N$ does not necessarily have to be aligned with image axes.

Our key insight is that a PVHV can be defined using two edge rays passing through inner blocker fragment $f$; we assume fragments have a square shape. The one edge ray $e(s, f)$ emanates from $s$ through $f$; see the thick blue line in Fig. 3(a). The other edge ray $e(n, f)$ emanates from $n$ through $f$. Also, for the adjacent outer blocker fragment $g$, the PVHV exists only when $f_2 < q_2$ (the subscript $z$ indicates the depth of a point).

Let $p$ be the 3D point of an incoming fragment to test. Then, we can find a line segment belonging to the PVHV at $z = p_2$ (the thick green line in Fig. 3(a)), which passes through $p$ and is parallel to $n - s$. Let the intersection of the line segment with $e(s, f)$ be $p'$. Then, a simple triangle similarity gives us the length of the line segment as:

$$R(p, f) = \left(\frac{p_2 - f_2}{f_2}\right)E, \quad \text{where} \quad p_2 > f_2. \quad (1)$$

This relationship resembles the Circle of Confusion (COC) used in DOF rendering [Potmesil and Chakravarty 1981], but is locally defined for $f_2$ instead of a global focal depth. For convenience, we let this length be the size of Local COC (LCOC). Then, a PVHV $H_s(N)$, encompassing all the novel views $N$, can be defined as:

$$H_s(N) = \{p \in W \mid p_2 > f_2, |p - p'| < R(p, f), (p - p') \cdot (n - s) < 0\}. \quad (2)$$

The last condition, $(p - p') \cdot (n - s) < 0$, ensures to exclude the fragment on the left side of $e(s, f)$, and for $f$ to be the inner occluder of $p$. When a fragment $p \in O_s(N)$, it is potentially visible from novel views, and needs to be captured in the multi-fragment rendering.
3.2.2 Point Novel View. A PVHV for a point novel view (the green region in Fig. 3(b)) is a truncated variant of that for linear views, which is cut by an additional edge \((e, n, q)\) (the thick red line in Fig. 3(b)). As a result, the sub-triangle (the orange region in the figure) is not in the PVHV.

It is possible to handle this case in theory, but this adds too much complexity in practice, because we have to find \(g\) as well as \(f\). For simplicity, we handle this using the same process as before for the linear views, since this can be conservatively included in a larger volume of the linear views. This merging scheme is similar in approach to occluder fusion [Schaufler et al. 2000; Wonka et al. 2000] or virtual occluders [Koltun et al. 2000] used in PVS.

3.2.3 Areal Novel Views. The last type can be defined by areal views that span spatially in a circle or rectangle. The former corresponds to the lens-based DOF rendering, and the latter, to the typical light field rendering. For the circular views, the PVHV can be the revolution of the 2D PVHV triangle of the linear views, and for the rectangular views, the PVHV has an oblique pyramidal shape that stacks mirrored rectangles from the inner blocker. In this work, we focus on the circular areal views.

Depending on the position of blockers, the opposite half of the views are irrelevant to forming PVHVs. In other words, half of novel views cannot see through the gap between \(f\) and \(g\) in this case, because \(f\) occludes them. The precise shape of the half cone can be defined by looking at the edge through fragment \(g\), but this requires an additional search around \(g\). Assuming a straight edge through \(g\) for simplicity, its planar cut makes the resulting PVHV an oblique truncated semicircular cone (Fig. 3(c)). This can still be identified when using the same equation, Eq. (2), for 2D views. In this case, we let \(h\) be the farthest view along the projection of \(f - s\) onto the plane of the novel views. We note that our implementation of EDP uses a bidirectional search employing the entire circular views, and does not find the edge shape through \(g\) in practice (see Sec. 4.1).

3.3 PVHV for Multi-Fragment Rendering

Our PVHV model can be applied to many of the multi-fragment rendering algorithms. When a single geometry pass captures fragments (e.g., A-buffer and k-buffer), we can cull redundant fragments after the pass. When the fragments are captured in order in multiple geometry passes (e.g., DP), we may use on-the-fly culling.

A reference brute-force algorithm for PVHV-based fragment culling per layer can use a two-pass implementation. Given all the captured fragments, we first find all the edge fragments (\(f\) in Fig. 3) and store their indices. Then, in the second pass, all the fragments are tested using Eq. (2) if any of the PVHVs of the (neighboring) edges includes their back-projections to the 3D space (\(p\) in Fig. 3). If a fragment passes the test, it needs to be kept. Otherwise, we discard the completely occluded fragment.

4 PVHV-BASED DEPTH PEELING

In this section, we first present our effective DP algorithm, which realizes our PVHVs for on-the-fly fragment culling. To meet real-time constraints for GPU-based DP, we propose a sampling-based approximate but efficient algorithm. Then, we present a procedure for multi-layer iteration, and fragment packing for storage efficiency.
Algorithm 1 Effective Depth Peeling

1: procedure LCOC(p₁, f₂) → screen-space LCOC (radius)
2: return ProjectToScreen(((p₂ − f₂)/f₂)E)
3: procedure EdgeExists(w, q) → connectivity test
4: if w.item ≠ q.item then return true
5: if w.z > (q.z + λ) then return true
6: return false
7: procedure InPVHV(p) → input fragment p (of depth p₂)
8: q ← p; q.z ← Z(pxy) → blocker q of depth q₂
9: R ← LCOC(p₂, q₂) → screen-space radius of LCOC
10: Ω ← GetRandomSamples(R) → a set of neighbors
11: foreach w ∈ Ω do
12: w.z ← Z(pxy + w) → sample depth
13: if EdgeExists(w, q) then return true → found PVHV
14: if w.z ≥ (q.z − Δ) then continue → in connected surface
15: if conservative then → bound extension
16: if InPVHV(pxy + w, p₂) then return true
17: else return true → conservative approximation
18: return false → fully hidden, included in no PVHVs

Fig. 5. Recursive bound extension. Given a point p and its projection q onto A, a sample w cast in Ω (made by q) finds q’ in B that is nearer than A. A test in Ω’ (made by q’) can find another PVHV formed by B and C.

Fig. 6. Propagation of PVHV-based culling applied to early layers. When EDP is applied to h ≤ 2 and a simpler culling (here, the umbra culling [Lee et al. 2010]) is applied to deep layers (here, h = 3), its culling efficiency is still similar to those generated only with EDP. The number of fragments (relative to the screen) are 0.0426, 0.0456, and 0.217 for (a), (b), and (c), respectively.

4.1.1 Implementation. Algorithm 1 shows a pseudocode for our EDP algorithm for a single geometry pass, which works for circular areal views. Given the previous-layer depth texture Z and the bound radius of novel views E, every fragment p is tested using InPVHV. We first assign the depth q₂ to the blocker q at the same screen-space location of p (line 8). Then, we collect screen-space random samples (line 10) in the circle of the radius of LCOC (line 9); we use Poisson-disk samples or Halton sequences. When an edge exists (using the differences of object IDs and depths; lines 4–5) for the sample (line 13), the fragment is in a PVHV; in other words, a novel view can look through the gap made by the edge. In this case, the test is terminated, indicating a PVHV is found. Otherwise, we do additional tests. If the depth difference is little (i.e., the sample in the connected surface), w is bypassed (line 14).

The lines 15–17 handle the other case (w.z < q.z − Δ). By default, we may skip the sample, because this does not form a PVHV. However, the sample can find another secondary inner occluder, and form additional PVHVs (see Fig. 5). We can recursively handle this case by extending bound at w (line 16); in GPU implementation (not supporting recursion), this can be done using a Halton sequence [Halton 1964] so that subsequent quasi-random samples handle extensions.

However, this recursion requires many excessive samples. We propose a simpler yet conservative approximation that considers such samples are potentially in PVHVs (line 17). Most of the unveiled fragments are found at the first inner occluder, and the extension hardly produces a difference. In our experiments, this adds marginal false positives (occluded but considered visible).

4.2 Multi-Layer Iteration

The PVHV-based peeling of layer h is always based on non-empty fragments only in layer h − 1. The first hidden layer (h = 1) uses fragments in the foreground image (h = 0; the visible layer rendered without peeling) as occluders. A deeper layer (h > 1) repeats the same way with respect to its previous layer (h − 1).

The multi-layer iteration of our EDP can follow what the typical DP pipeline does. However, most of the large-area occluder fragments are culled in the first hidden layer (h = 1), and some of the remainders are culled in the second hidden layer (h = 2). Hence, the PVHV-based culling for deeper layers (e.g., h > 2) gains little, but adds non-trivial overhead for the visibility tests.

We, therefore, apply EDP only for frontmost layers (e.g., h ≤ 2), and use a simpler pixel-based culling to deeper layers. To this end, we use “Umbra culling”-based DP (UDP) [Lee et al. 2010], which considers a back-projected pixel extent as an occluder and culls fragments in its umbrae formed by novel views. The EDP-driven initial culling still greatly affects deeper layers, because the effect of culling in the earlier layers is propagated to deeper layers (see Fig. 6). Nevertheless, EDP only with the frontmost layers (h ≤ 2) already performs similarly with simpler DP techniques peeled for deeper layers; our experiments show the effects of this hybrid scheme.
4.3 Fragment Packing
By default, our EDP consumes non-trivial fixed-size memory that scales exactly with the number of layers and the screen resolution. However, in our case, we can optionally reduce memory consumption by tightly packing sparse valid fragments.

When this Packed EDP (PEDP) is used, we use a linked list on GPU [Yang et al. 2010], which dynamically stores fragments in a node buffer with a head texture (having pointers to the foreground fragment node). The packing is post-performed after every layer is peeled, as per-layer process. This per-layer packing avoids keeping the entire layers in memory, and requires only two working textures, one for input $h-1$ and the other for output $h$; they alternate at every layer iteration in the ping-pong fashion. In Sec. 5, we demonstrate how much memory consumption PEDP can reduce.

5 RESULTS
This section reports our experimental evaluation of PVHVs applied to EDP (DP integrating PVHV-based fragment culling) for Depth-of-Field (DOF) rendering as a multi-view warping application.

5.1 Method
5.1.1 Platform and Scenes. Our algorithm was implemented in OpenGL 4.6 on an Intel Core i9-10900 2.8 GHz and NVIDIA GeForce GTX 3090. All the tests used the full-HD resolution (1920×1080). Three scenes are used for experiments (Fig. 7). The Ruins scene (RU) has high depth complexity with a moderate number of objects. The Safari scene (SF) is of moderate depth complexity with a moderate number of objects. The Satellites scene (ST) is challenging in terms both of depth complexity and number of objects.

5.1.2 Layer Notations. We let $L$ be the number of hidden layers, and $h$ be the index of layers; for instance, given $L=2$, $h=0$ indicates the visible foreground layer, and $h=1$ and $h=2$ indicate the first and second hidden layers, respectively. We note that the foreground rendering (for $h=0$) is excluded in the comparison of peeling and warping, since it is irrelevant to hidden layers and is commonly required to any of DP techniques.

5.1.3 Configuration of EDP. As mentioned in Sec. 4.2, we apply EDP only to the first two hidden layers ($h \in [1, 2]$), and use UDP [Lee et al. 2010] for $h > 2$. A fragment encodes color, depth, and item index using RGBA32F format. EDP-specific constants include: (1) the depth-difference threshold $\Delta$; (2) the number of backward-search samples $|\Omega|$. We use $\Delta=0.002$ (in the normalized linear scale) by default, but show its effects on quality. $|\Omega|$ is set to 14 by default; we also show the effects of different $|\Omega|$ values.

5.1.4 Previous DP Techniques. We consider the original GPU-based DP [Everitt 2001] as a Baseline DP (BDP). BDP uses little threshold (0.0005 in normalized linear depth scale) to avoid depth fighting. We also compare ours with the original UDP [Lee et al. 2010].

5.1.5 Peeling Errors. Peeling errors are measured by false negative pixels in the novel-view reference rendering. Every reference pixel
Table 1. Comparison of peeling efficiency of the full-layer EDP (ours) and UDP [Lee et al. 2010], evaluated as relative reduction of the number of hidden fragments ($h > 0$) against BDP [Everitt 2001]; higher is better.

<table>
<thead>
<tr>
<th>Scene</th>
<th>E (mm)</th>
<th>Ruins</th>
<th>Safari</th>
<th>Satellites</th>
</tr>
</thead>
<tbody>
<tr>
<td>BDP/EDP</td>
<td>0.1</td>
<td>5.5x</td>
<td>4.4x</td>
<td>3.0x</td>
</tr>
<tr>
<td>BDP/UDP</td>
<td>2.2</td>
<td>1.7x</td>
<td>1.3x</td>
<td>1.1x</td>
</tr>
</tbody>
</table>

Table 2. Memory consumption and efficiency of hidden fragments of PEDP, and hidden layers of EDP and UDP [Lee et al. 2010] in the full-layer peeling. The memory efficiency is evaluated as relative reduction against BDP [Everitt 2001]; higher is better. The numbers in parentheses are the number of the peeled hidden layers. The bold face indicates the best values.

<table>
<thead>
<tr>
<th>Scene</th>
<th>Memory (MB)</th>
<th>Memory Efficiency</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>PEDP</td>
<td>EDP</td>
</tr>
<tr>
<td>RU 50</td>
<td>17.7</td>
<td>316.4 (10)</td>
</tr>
<tr>
<td>SF 100</td>
<td>16.4</td>
<td>221.5 (07)</td>
</tr>
<tr>
<td>ST 1000</td>
<td>9.0</td>
<td>94.9 (03)</td>
</tr>
</tbody>
</table>

is reprojected to the peeled layers. If no potential source pixels are found in the layers, we tag the target pixel as erroneous.

5.1.6 Application: Multi-View Warping in DOF Rendering. We demonstrate our EDP in multi-view DOF rendering as a warping application. The DOF rendering actually has nothing to do with PVHV extraction, but defines the maximum extent $E$ of novel views. The novel views are defined as lens samples in the circular lens of the radius $E$ (Fig. 3). For the quality evaluation, we use the accumulation buffering [Haeberli and Akeley 1990] (using 4096 lens samples) as a reference. The evaluation uses still shots, as shown in the figures; additional evaluations, showing averages over animated camera sequences, are found in the accompanying video clip.

The final DOF image is generated by accumulating the novel-view images warped for each lens sample. The per-fragment warping uses compute shader-based forward warping [Cichocki 2017; Yu et al. 2010]. Every peeled fragment is reprojected to a pixel position at a novel view and tested against a custom depth buffer. Conflicts occur when multiple fragments map to the same location, and so, the depth is atomically tested. A seam can be made in the warping, when a source texel slightly enlarges in the novel view [Yu et al. 2010]. We handle this using subpixel offsets (we use four subpixel samples); to our knowledge, there is no simpler solution to avoid these seams in the forward warping.

Another practical issue is the corrupt periphery of warping for large view displacements. This results from the lack of source pixels that can map to the target positions at the image boundary. To handle this, we extend the field of view for DP to perform wide-angle projection as used in [Reinert et al. 2016]; in this work, we scale the screen dimension by a constant factor (1.2).

5.2 Peeling Efficiency and Memory Consumption

5.2.1 Peeling Efficiency. We first assess peeling efficiency in terms of number of hidden fragments in the full-layer DP to show how many hidden fragments our EDP can reduce with respect to the traditional DP (BDP) as a baseline. We repeat peeling until no fragments are generated, and count the number of hidden fragments for BDP, UDP, and EDP; foreground fragments (in $h = 0$) are excluded here. Our EDP utilizes UDP for $h = 2$. The peeling efficiency is measured as a relative reduction to BDP; for example, $2\times$ (that is, 2:1) indicates the half of fragments are peeled with respect to those peeled by BDP.

Fig. 7 visually compares the reduction of fragments in the first hidden layer (of $h = 1$), which is the most important for warping. EDP drastically reduces the number of fragments than BDP and UDP do. The resulting fragments are already sparse. Most fragments behind large occluders are discarded well, and fragments neighboring to edges (important for warping) are maintained. Empty fragments in the hidden layer do not yield further fragments at the location, and thus, peeling at deeper layers becomes more efficient.

5.2.2 Memory Consumption. Memory consumption matters for constrained environments. The memory consumption of non-packed schemes, including BDP, UDP, and EDP, scales with the screen resolution (here, 1920×1080) and the number of hidden layers (i.e., deepest layer where any fragments exist). A single layer of RGBA32F format consumes 31.6 MB. Since the full-layer peelings of BDP, UDP,
and EDP do not generate the same number of layers, their memory consumptions are different: given the same number of layers, the memory consumptions should be the same.

In contrast, the memory consumption of PEDP (EDP with fragment packing) scales directly with the peeling efficiency, since the packing removes empty pixels. A single node of the linked list of PEDP consumes 16 bytes in the RGBA32F format (8, 4, and 4 bytes for color, depth, and indexed pointer to the next node, respectively). The consumption of the entire linked list is multiplied with the number of fragments and subtracts 31.6 MB to exclude the foreground layer. In addition, PEDP requires a head texture of the screen resolution (7.9 MB), which holds 4-byte indexes to the first nodes per pixel.

Table 2 compares the memory efficiencies, which are measured as relative reductions against BDP. We again note that our EDP utilizes UDP for $h > 2$. PEDP packs the same number of fragments that EDP generates. The memory efficiency of our PEDP is very significant (here, $[25.1, 166.1] \times$). The numbers of peeled layers of EDP and UDP are not as low as PEDP, but are reduced well; EDP still performs better than UDP ($[1.0, 1.7] \times$) against UDP.

### Table 2

<table>
<thead>
<tr>
<th>E (mm)</th>
<th>Views</th>
<th>Foreground ($h = 0$)</th>
<th>BDP ($h &gt; 0$)</th>
<th>UDP ($h &gt; 0$)</th>
<th>EDP ($h &gt; 0$)</th>
<th>Speedup of DP+W ($h &gt; 0$)</th>
</tr>
</thead>
<tbody>
<tr>
<td>RU</td>
<td>100</td>
<td>256</td>
<td>45.152±0.003</td>
<td>46.104±0.004</td>
<td>52.076±0.005</td>
<td>1.0</td>
</tr>
<tr>
<td></td>
<td>1024</td>
<td></td>
<td>67.6</td>
<td>70.8</td>
<td>72.8</td>
<td>5.0</td>
</tr>
<tr>
<td>SF</td>
<td>100</td>
<td>256</td>
<td>48.341±0.030</td>
<td>49.209±0.034</td>
<td>52.180±0.036</td>
<td>4.0</td>
</tr>
<tr>
<td></td>
<td>1024</td>
<td></td>
<td>60.6</td>
<td>63.0</td>
<td>65.4</td>
<td>4.0</td>
</tr>
<tr>
<td>ST</td>
<td>1000</td>
<td>256</td>
<td>34.767±0.015</td>
<td>34.870±0.015</td>
<td>35.8</td>
<td>4.0</td>
</tr>
<tr>
<td></td>
<td>1024</td>
<td></td>
<td>64.1</td>
<td>64.8</td>
<td>65.8</td>
<td>4.0</td>
</tr>
</tbody>
</table>

Table 3 also shows peeling costs of the hidden layers for the equal-quality layers; the baseline costs of peeling and warping for the foreground layer ($h = 0$) are also shown in the table. When peeling a single layer, EDP is costlier than BDP and UDP, but its fewer layers having the equal qualities allow much lighter multi-layer peeling.

The cost of hidden-layer warping of EDP is much lower than those of UDP and BDP due to the fewer layers and fragments. The speedup factors of EDP in the combined costs (for DP and warping of hidden layers) are also much higher. Here, ours significantly outperforms BDP and UDP in all the cases. The speedup factors are very high, and range in $2.3-3.4 \times$ and $1.6-2.9 \times$ with respect to BDP and UDP.

When PEDP (EDP with packing) is used, the packing itself adds slightly more cost. For the two hidden layers ($L = 2$), the packing takes 0.6–0.8 ms. The combined cost (of the peeling, packing, and warping for hidden layers) are still much lower than non-packed BDP and UDP. The average speedup factors of PEDP against BDP and UDP are $2.2 \times$ and $1.8 \times$ for all the combinations of $E$ and the scenes. PEDP precludes empty fragments in the warping, but additional dereferencing to the first pointer in the head texture is required in the warping. Also, the random memory access during the node traversal can be cache-unfriendly. Hence, PEDP is likely to be slightly slower than EDP. Nevertheless, PEDP-based warping still outperforms BDP and UDP, and is a good alternative to BDP and UDP, as well as EDP, in resource-limited environments.

5.4 Peeling Accuracy

Our PVH algorithm is not approximate, but our EDP implementation is approximate and may produce artifacts (i.e., missing warped

```markdown
Table 4. Comparison of errors (false-negative pixels in the reference) when only a single hidden layer is peeled ($L = 1$). The numbers are relative fractions to the screen resolution. The best values are indicated in the bold face.

<table>
<thead>
<tr>
<th>BDP</th>
<th>UDP</th>
<th>EDP</th>
</tr>
</thead>
<tbody>
<tr>
<td>RU</td>
<td>0.00167</td>
<td>0.00156</td>
</tr>
<tr>
<td>SF</td>
<td>0.00142</td>
<td>0.00134</td>
</tr>
<tr>
<td>ST</td>
<td>0.00760</td>
<td>0.00707</td>
</tr>
</tbody>
</table>
```

Fig. 9. Comparison of errors (areas indicated as the green color) of warping for a single novel view ($(-0.468 0.868)$ in a unit circle here) for RU scene.

5.3 Equal-Quality Performance of Peeling and Warping

The quality and performance of DP and warping (for DOF rendering) depend both on the numbers of hidden layers and fragments. Hence, we compare performance in equal-quality settings, and control the number of layers to find near-equal qualities for different DP methods. Table 3 shows the number of layers required for the same qualities. In these cases, our EDP requires only two hidden layers to produce high-quality DOF rendering for the complex scenes (see Fig. 8 for the rendered images). Given the equal qualities per scene, our EDP can reduce the number of effective layers by up to 2/2/3 (RU/SF/ST) for BDP and 3/2/2 (RU/SF/ST) for UDP.
Fig. 10. Effect of the number of backward-search samples \(|\Omega|\) of EDP for the first hidden layer \((h = 1)\) and the warping generated with two hidden layers \((L = 2)\). The novel view is generated at \((-0.35, -0.48)\) using \(E = 100\) mm and \(\Delta = 0.002\). The numbers of the erroneous pixels here are 9637, 9, 0, and 0 for \(|\Omega| = \{2, 4, 8, 16\}\) at the full-HD resolution, respectively.

Fig. 11. Comparison of the worst-, balanced-, and best-case scenarios in the use cases of our EDP. The novel views are sampled at \((-0.52, 0.76)\) and \((-0.66, -0.71)\) for RU and ST scenes, respectively. A smaller \(\Delta\), a larger number of \(|\Omega|\), a smaller \(E\), and a larger number of hidden layers \((L)\) result in higher quality; the best examples are shown in (c). The most balanced settings in terms of performance and quality are found in the middle column (b).

pixels) in complex scenes. The sources of errors, common to any DP (including BDP), can be as follows. First, any DP may lose surfaces orthogonal to a view plane. Second, back-face culling may miss faces inverting topology with large view displacements. The additional source of errors particular to ours is the limited precision of surface connectivity tests. Few adjacent disjoint surfaces can be considered connected, and insufficient samples may not find existing edges.

Nonetheless, the erroneous pixels are negligible and are sparsely found. Our experiments found that the average fractions of erroneous pixels of the full-layer EDP are only 0.000014, 0.000098, and 0.000027 for RU, SF, and ST, respectively. Those of BDP are 0.000013, 0.000093, and 0.000024. The differences of EDP from BDP are only 1, 11, and 5 pixels in the full-HD resolution.

Table 4 shows errors in the warping when only a single hidden layer \((i.e., h = 1\) for \(L = 1\)) is peeled (with 14 samples). The exhaustive full-layer DP might be inviable in practice. In such cases, capability for the early capture of important fragments is crucial. It is shown that our EDP captures important fragments much earlier than BDP and UDP; BDP and UDP require peeling more layers for correct warping (see Fig. 9 for visual comparison).

\(\Delta\) of EDP trades errors for peeling efficiency, but is insensitive to scenes, because our additional surface connectivity metric based on the item buffer helps to avoid erroneous cases to a large extent. The best balance is found at \(\Delta = 0.002\) (our default setting).

Fig. 10 shows the effect of the number of backward-search samples \(|\Omega|\). When samples are not enough (here, 2 and 4 samples), PVHVs are not found despite the presence of edges. More samples (here, 8 and 16 samples) lower the chances of missing existing edges, and reconstruct surfaces correctly in the warping. In our experiments, \(|\Omega|\) has little impact in medium-complexity scenes, when more than 14 samples (our default setting) are provided. Complex scenes and a larger \(E\) may require more samples to not miss visible fragments.

The combined effects of \(\Delta\) and \(|\Omega|\) for the different values of \(E\) and \(L\) are shown in Fig. 11. The selected configurations exemplify the worst, balanced, and best cases that our EDP can generate. In general, a larger \(\Delta\), a smaller number of \(|\Omega|\), a larger \(E\), and a smaller number
of hidden layers are more likely to miss visible pixels. The worst-case examples with non-trivial errors are shown in the left column of the figure. The errors are often found in complex junctions, where our edge detection may fail. When a sufficient number of samples and layers with a small Λ (0.002; our default setting) are provided (the right column in the figure), the errors are hardly found in our experiments. A balanced setting is shown in the middle column, which requires a small number of layers and produces subtle errors. Whereas pixels that should be visible are not extracted, their impact on memory and performance is not critical, since usually small portions of fragments are affected. The most important factors for high performance and memory efficiency are |Ω| and the number of layers; hence, keeping them small is crucial in practice. Nevertheless, when integrating many multiple views, the errors are diluted and hardly visible.

Lastly, we report the effect of increasing novel-view displacements (E) (Fig. 12). When E becomes larger, more fragments in PVHVs are revealed. At some points, most of the fragments are revealed as peeled by BDP. Naturally, this leads to a slowdown of the entire pipeline, because more fragments have to be processed in the warping. Nonetheless, the peeling costs do not make a big difference.

The differences of per-layer costs between E × 1 and E × 16 are only 0.28, 0.15, and 0.56 ms for RU, SF, and ST, respectively. On the other hand, the warping performance is degraded (Fig. 13), but scales well with the increasing E values. For the moderate values of E (up to {400, 400, 4000} mm for RU, SF, and ST scenes, respectively), the performance degrades linearly. When E becomes unrealistically large (E > {400, 400, 4000} mm for RU, SF, and ST scenes, respectively), the degradation patterns are sub-linear, because many fragments are likely to be bypassed in the warping when the target positions of their warping extend beyond the image boundary.

6 CONCLUSION AND DISCUSSIONS
We presented the model and algorithm of Potentially Visible Hidden Volumes (PVHVs) for multi-view image warping. We realized PVHVs as an efficient on-the-fly fragment culling in DP. Our experiments show that our Effective DP (EDP) can greatly reduce hidden fragments, while keeping higher quality given the limited number of layers and improving warping performance.

As already mentioned, our PVHV algorithm is not approximate, but our implementation of EDP may produce artifacts in complex scenes. While the errors are marginal, we can further suppress the errors using stricter connectivity tests. For example, a lower Λ value and many search samples can be used, but these may trade peeling efficiency (i.e., performance) for quality.

We demonstrated the utility of PVHVs and EDP in DOF rendering. Different types of multi-view rendering (e.g., multi-view soft shadows, light-field rendering, and temporal multi-view warping) can also benefit from the PVHVs and EDP. Also, different types of multi-fragment DOF rendering can benefit from our solution. For examples, the multi-layer approaches based on the image-space ray tracing [Lee et al. 2009, 2010] and multi-layer filtering [Selgrad et al. 2015] can use fewer layers for the similar quality and thereby achieve higher rendering performance. Further studies on additional use cases and applications are encouraged.

ACKNOWLEDGMENTS
Correspondence concerning this article can be addressed to Sungkil Lee. The reference implementation of Algorithm 1, written in OpenGL Shading Language, is available at https://github.com/cgskku/pvhv.
REFERENCES


