Efficient Hybrid Image Warping for High Frame-Rate Stereoscopic Rendering

Andre Schollmeyer, Simon Schneegans, Stephan Beck, Anthony Steed and Bernd Froehlich

(a) LOD point cloud rendering of a scanned rock scene

(b) Rendering with highest LOD quality (20Hz)

(c) Rendering with low LOD quality (60Hz)

(d) Rendering with high quality and image warping (60Hz)

Fig. 1. Our hybrid image warping strategy provides effective high frame rates with high image quality. (a) Shows a high-quality rendering of a level-of-detail (LOD) point cloud. (b) A zoom-in of this image. At this level of quality, conventional stereoscopic rendering is quite slow. (c) Shows the level of quality achievable at 60Hz with conventional rendering. A much lower level of detail must be used. (d) With our method, 60Hz is achievable at a higher level of detail. Even with minor warping artifacts, the image quality is significantly better.

Abstract—Modern virtual reality simulations require a constant high-frame rate from the rendering engine. They may also require very low latency and stereo images. Previous rendering engines for virtual reality applications have exploited spatial and temporal coherence by using image-warping to re-use previous frames or to render a stereo pair at lower cost than running the full render pipeline twice. However these previous approaches have shown artifacts or have not scaled well with image size. We present a new image-warping algorithm that has several novel contributions: an adaptive grid generation algorithm for proxy geometry for image warping; a low-pass hole-filling algorithm to address un-occlusion; and support for transparent surfaces by efficiently ray casting transparent fragments stored in per-pixel linked lists of an A-Buffer. We evaluate our algorithm with a variety of challenging test cases. The results show that it achieves better quality image-warping than state-of-the-art techniques and that it can support transparent surfaces effectively. Finally, we show that our algorithm can achieve image warping at rates suitable for practical use in a variety of applications on modern virtual reality equipment.

Index Terms—Image warping, stereoscopic rendering, transparency warping, a-buffer ray casting, image warping strategies, surface estimation quadtree

1 Introduction

The real-time generation of realistic imagery remains a considerable challenge for any rendering engine. Advancements in graphics hardware are often compensated by an increasing demand for highly detailed models, sophisticated shading effects and high display resolutions. Furthermore, immersive 3D displays such as head-mounted displays (HMDs), 3D monitors or stereoscopic projection systems are on the verge of becoming standard consumer products. For these displays, high frame rates and low latency are essential requirements to prevent simulator sickness and provide smooth interaction. For example, the Oculus Rift CV1 is recommending a consistent 90Hz rendering rate. They also need imagery to be generated for both eyes. As a result, trade-offs between visual quality and high frame rates become often necessary.

- *Andre Schollmeyer, Stephan Beck and Bernd Froehlich are with the Virtual Reality Systems Group, Media Faculty, Bauhaus-Universität Weimar. E-mail: <firstname>.<lastname>@uni-weimar.de.*
- *Simon Schneegans is with German Aerospace Center (DLR), Braunschweig. E-mail: Simon.Schneegans@dlr.de.*
- *Anthony Steed is with University College London. E-mail: a.steed@ucl.ac.uk.*

Manuscript received xx xxx. 201x; accepted xx xxx. 201x. Date of Publication xx xxx. 201x; date of current version xx xxx. 201x. For information on obtaining reprints of this article, please send e-mail to: reprints@ieee.org. Digital Object Identifier: xx.xxxx/TVCG.201x.xxxxxxx

Recent HMD frameworks minimize the latency by predicting the user's head movements just before display and update the already rendered image by a 2D warp[1]. This 2D transformation would be sufficient for pure eye rotations, but it also works well for head rotations because they result in only small positional changes of the eyes. For position changes of the head, 3D image warping involving the depth buffer is necessary [15]. 3D warping has also been used for stereoscopic image generation [6]. However, most existing 3D warping approaches do not scale well with increasing image resolution and are prone to visual artifacts. Furthermore, existing warping approaches for semi-transparent surfaces [12] are limited to a single layer.

In this paper, we present novel techniques to increase the scalability, applicability and visual quality of 3D image warping for stereoscopic displays. We start by generating an image using an existing deferred rendering engine which stores opaque fragments in a G-Buffer [21] and transparent fragments in per-pixel lists of an A-buffer [11]. For the G-Buffer, an adaptive grid is generated based on the curvature and continuity of the contained depth image. For the A-Buffer, a min-max quadtree is built to accelerate backward warping by ray casting. The current stereoscopic views are generated from a reference image by combined forward and backward warping using these data structures. The grid and the min-max quadtree are both independent of the warp direction and can be used multiple times. Potential artifacts are reduced by a novel hole-filling strategy.

[1] https://developer.oculus.com/blog/asynchronous-timewarp-examined/

The main contributions of our approach can be summarized as follows:

- A hybrid warping approach combining grid reprojection for opaque pixels and ray casting of semi-transparent fragments
- A GPU-based adaptive grid generation for 3D warping which results in fewer primitives than other state-of-the-art approaches
- An efficient A-Buffer ray casting accelerated by a min-max quadtree which is built on-the-fly
- A GPU-based depth-aware, low-pass filter for hole filling which achieves higher quality than existing algorithms

We demonstrate our algorithms for practical virtual reality scenarios by extending an open-source rendering engine. We provide two warping strategies, one that is best suitable for mostly static scenes and one that works better for dynamic scenes. Our implementation shows that the performance of the warping scales well with high resolutions due to the adaptive warping grid. Furthermore, it proves that image warping is not limited to opaque geometry, but can also be combined with per-fragment programmable transparency. An evaluation shows that our approach produces better results than other state-of-the-art approaches and may improve performance as well as latency. In addition, warping in combination with an output-sensitive rendering system may also be used to significantly improve visual quality while maintaining the same frame rate as conventional rendering, as shown in Figure 1. A user study confirms that in some scenarios users strongly prefer stereoscopic warping over conventional stereoscopic rendering while warping artifacts go largely unnoticed.

2 Related Work

Image warping, that is applying geometric transformations to a source image, is a mature field with several application areas. These transformations often use additional per-pixel information such as depth or motion. The resulting target image may appear as if it was created for another perspective. In general, warping is neither injective nor surjective, i.e. the target image may contain artifacts caused by holes and folds. A survey of rendering systems exploiting temporal coherence by image warping was given by [22]. Furthermore, image warping has been used to generate post-processing effects such as motion blur and depth of field [16].

2.1 Warping of opaque objects

Existing warping algorithms can be categorized by their data-access pattern: forward-warping algorithms are based on data scattering and backward-warping approaches are based on data gathering.

The most common data-scattering approach is to render one point for each pixel in the source image [15] which was also shown for multiple layered depth images [24]. A problem of this approach is that the calculated target locations are usually not at discrete pixel locations and coloring the pixel closest to the calculated position will lead to aliasing artifacts. Therefore, techniques such as point splatting [30] were proposed and have been widely used in the context of image warping. However, transforming each pixel separately does not scale well with increasing image resolutions. Since adjacent pixels from the source image often keep their relationship after warping, Chen et al. [4] proposed to warp such areas as blocks. This idea was improved by Didyk et al. [6]. Their GPU-based implementation repeatedly subdivides a coarse screen-space grid until all pixels of the grid cells have a maximum allowed depth disparity. The resulting grid is transformed and rasterized in the destination image space. For opaque objects, we follow the idea of using an adaptive grid but provide more efficient generation schemes.

For data gathering approaches, suitable color information is retrieved from a single or multiple source images [31] by an iterative search. An advantage of this method is that no z-buffering is required because contributing pixels are gathered and composited in one step. Bowles et al. [2] proposed a search based on a fixed-point iteration. Motivated by an in-depth analysis of its convergence behavior, they enhanced their system by using adaptive grid warping to find appropriate iteration starting points. Another data gathering approach is ray casting. Peek et al. [18] used basic ray casting into the depth buffer to perform translational warping for latency reduction of HMDs.

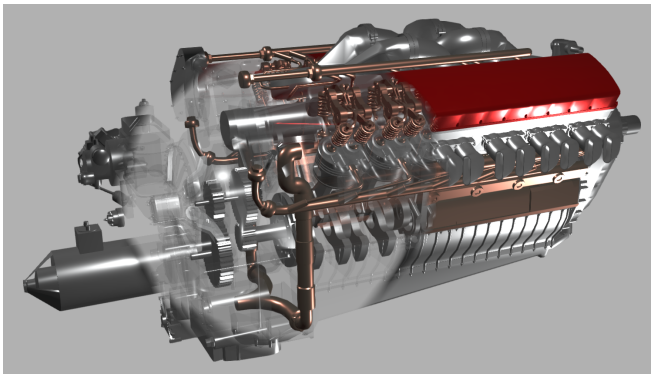


Fig. 2. Visualization of an engine model. Inner parts can be explored using a see-through lens which requires programmable transparency.

Both, data scattering and data gathering are used by our system. Data scattering allows for very quick warping of opaque surfaces while data gathering will be used to composite semi-transparent image information.

2.2 Hole Filling

In many cases, there is not sufficient information in the source image to correctly determine the color of each output pixel. Many strategies have been proposed either to prevent or fill the resulting holes.

The most common preventive hole-filling approach is to stretch neighboring texture information over the holes [14]. This results in artificial geometry, so-called "rubber sheets", spanning the gap between foreground and background. While this popular method is cost-efficient, it introduces noticeable artifacts especially in the presence of high frequencies in the depth buffer. In order to reduce these artifacts, several works perform a low-pass filtering on the depth-buffer [10, 19, 20, 32].

Reactive hole-filling methods try to fill holes and therefore relate to image reconstruction [17]. For tiny holes, simple inpainting methods may be sufficient, for instance, choosing a random neighboring pixel color [2]. For larger holes, it is possible to extend the boundary color of a hole in epipolar direction which has similar results as the preventive rubber sheet approach. Other research focuses on efficient hole-filling strategies based on ray casting [1]. However, for complex scenes, performing ray casting for scattered pixels may represent a potential performance bottleneck. Therefore, we suggest a novel reactive hole-filling algorithm.

2.3 Warping of semi-transparent objects

Many modern rendering engines are based on deferred shading [21]. The color and geometry information are stored in offscreen render targets (G-Buffer) which makes the integration of a warping stage straightforward. If support for transparent objects is required, warping becomes non-trivial. Blending transparencies before warping results in visual artifacts, as shown in Figure 7. Works extending the G-Buffer are limited to a single layer of semi-transparent objects [12]. A potential solution would be to render and blend transparent objects for each eye separately.

However, some systems support programmable transparency [23] which allows for the implementation of advanced 3D interfaces, e.g. see-through techniques [28] as shown in Figure 2. In such a system, the opacity of an object is computed on a per-fragment basis. All opaque pixels are stored in a G-Buffer, while all transparent fragments are routed into an A-Buffer [3] for later compositing. In this case, re-rendering transparent objects would usually require re-rendering large parts of the scene which is, of course, not suitable.

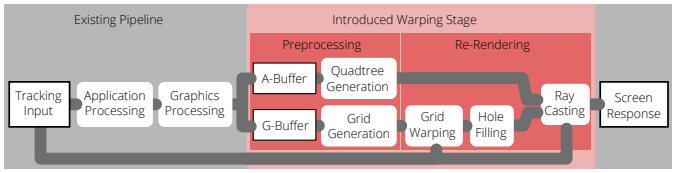


Fig. 3. The warping follows the application processing and the regular rendering. It is separated into two parts: A warp-direction independent preprocessing stage and the re-rendering which performs the actual warping based on the latest tracking data as additional input.

The A-Buffer is a versatile data structure for storing multiple semi-transparent fragments per pixel. The generation of this data structure has been presented before; however, nothing has been published regarding warping of the A-Buffer.

3 System Overview

The proposed 3D warping approach is added to an open-source VR-system which implements a deferred rendering pipeline and supports also programmable transparency [Schollmeyer2015]. Opaque pixels are stored in a G-Buffer while transparent fragments are stored in an A-Buffer. These two buffers form the basis of the warping stage, which needs to accomplish three major tasks: a 3D depth image warp of opaque pixels, the hole filling and the reprojection of transparent fragments. The costs of these tasks are highly dependent on the image resolution. We perform preprocessing to generate appropriate data structures for accelerating re-rendering. The conceptual integration of such a warping stage into an existing rendering pipeline is shown in Figure 3.

4 Warp Preprocessing

As described in the system overview, the results of the application-defined rendering pipeline are an A-Buffer and a G-Buffer. These two buffers are used to build the following data structures: an adaptive warp grid for opaque pixels and a min-max quadtree storing depth values for transparent fragments. Both are used to accelerate the subsequent re-rendering. In particular, the adaptive grid is used for 3D image warping, while the min-max quadtree minimizes the costs for the ray casting. However, they do not depend on the warp direction and thus may be reused multiple times, e.g for both eyes or for multiple frames in an asynchronous warping system.

4.1 Adaptive Grid Generation

The main goal of this stage is to find blocks of adjacent pixels that belong to the same almost flat and connected surface patch which can therefore be safely warped together without producing holes and geometric distortions. At first, we filter the depth buffer in order to find such surface patches. To allow for parallel processing, the source image's depth buffer is divided into tiles of a certain size, e.g. 32x32 pixel. For each tile, a surface-estimation quadtree is constructed bottom-up. A node in the quadtree only contains information about the connectedness and flatness of the corresponding area. Didyk et al. [6] derive connectedness information from the minimum and maximum depth values of all represented pixels stored in a quadtree. However, their approach leads to over-tessellation on surfaces which are connected, but tilted with respect to the camera, as indicated in Figure 4(a). In contrast, we provide three reduction strategies to construct a surface-estimation quadtree which also detects surfaces that are slanted in view space: *cross-kernel surface reduction*, *partial cross-kernel surface reduction* and *irregular grid reduction*.

In the following, we will elaborate on these three strategies and then we will describe how to build an adaptive grid from this tree.

4.1.1 Cross-Kernel Surface Reduction

Instead of storing and comparing absolute depth values or disparities, this estimate stores in each quadtree node whether all leaves below this node form a connected surface. For leaf nodes, the four represented pixels are assumed to be part of a larger flat surface patch if each of them is collinear with its adjacent pixels in 3D space. This is verified by a set of collinearity tests $L : \mathbb{R}^3 \rightarrow \mathbb{B}$. Each collinearity test is based on the change of differences between the pixel's depth d_i and its neighbors which is computed as follows:

$$f(d_{i-1}, d_i, d_{i+1}) = |(d_{i-1} - d_i) - (d_i - d_{i+1})| \tag{1}$$

Slightly curved surfaces are accounted for by introducing an ε-tolerance. If f is within this ε-tolerance, the three adjacent pixels are considered approximately collinear. Thus, a collinearity test L is defined as:

$$L(d_{i-1}, d_i, d_{i+1}) = \begin{cases} true & \text{if } f(d_{i-1}, d_i, d_{i+1}) < \varepsilon \\ false & \text{otherwise} \end{cases} \tag{2}$$

The cross-kernel surface reduction performs two collinearity tests for each of the four pixels and the corresponding neighbors in horizontal and vertical direction, as shown in 5(a). If all eight tests are positive, the four pixels are assumed to form a surface patch.

For all positive tests, the maximum deviation from collinearity is stored as an estimation error for curved surfaces and accumulated to higher levels.

The information of whether four pixels form a connected surface can be safely propagated to higher levels of the quadtree. If all four children of an inner node are part of a connected surface, they are assumed to be part of the very same surface, because neighboring patches used overlapping pixel pairs to check for collinearity. The resulting quadtree is used to generate an adaptive tessellation in screen space, as shown in Figure 4(b). At edges, however, the tessellation may be too fine because the depth discontinuity between the edge and the background is propagated to the inner nodes. This over-tessellation can be reduced using the partial cross-kernel reduction.

4.1.2 Partial Cross-Kernel Surface Reduction

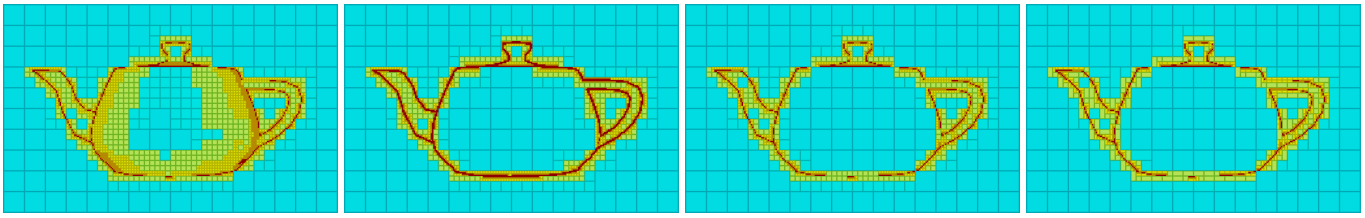
In order to detect surfaces close to depth discontinuities, the shape of the kernel has to be adaptive. Therefore, this reduction strategy assumes a surface if a group of four pixels is collinear to at least two adjacent sides, as indicated in Figure 5(b). Using this improved kernel, it is possible to detect connected surfaces which touch a depth discontinuity. However, it becomes non-trivial to propagate this information to higher levels of the quadtree. If two adjacent four-pixel groups form a surface each, it would still be possible that there is a discontinuity between them.

Therefore, each quad-tree node uses multiple bits to encode the connectedness characteristics. Each node separately encodes its connectedness to its left, right, top and bottom neighbor. In addition, storing this information to its four diagonal neighbors can improve the warp quality, as described in Section 5.1. Compared to the cross-kernel reduction, eight additional bits encode the partial connectedness of each node to its neighbors. Each bit is set if the adjacent pixels in the corresponding direction are collinear in 3D space. Using this information, it is possible to propagate surface information to higher levels in the quadtree. Each inner node forms a surface if its children are each part of a surface and they are connected in the corresponding directions. If that is the case, the node is marked as belonging to one surface and the connectedness bits of all children are merged by a logical *and*.

An example quadtree generated by this reduction is depicted in Figure 4(c). Far fewer primitives are generated which speeds up the actual 3D warping process.

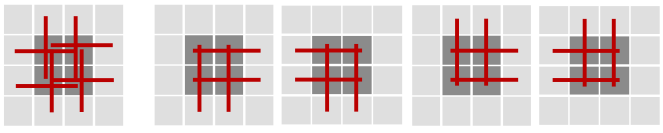
4.1.3 Irregular Grid Reduction

While the partial cross-kernel reduction already yields much better results, the number of cells generated can be reduced even further. The idea is to relax the constraint that each node of the quadtree has to have exactly four children: The irregular grid will also generate rectangular cells with an aspect ratio of $2:1$ in the following way. When four tree nodes are merged to create a new node on the next level, a merge



(a) Didyk et al. $\approx 9.3k$ (b) Cross-kernel $\approx 12.7k$ (c) Partial cross-kernel $\approx 6.7k$ (d) Irregular grid $\approx 4.3k$

Fig. 4. Depth-buffer quad trees for four different collinearity estimation algorithms and the corresponding number of primitives. The approach of Didyk et al. (a) tends to split inclined surfaces unnecessarily, while the cross-kernel estimation (b) leads to over-tessellation close to depth discontinuities. The partial cross-kernel estimation (c) yields much better results. The number of generated primitives can be decreased significantly if rectangular cells are allowed (d).



(a) (b)

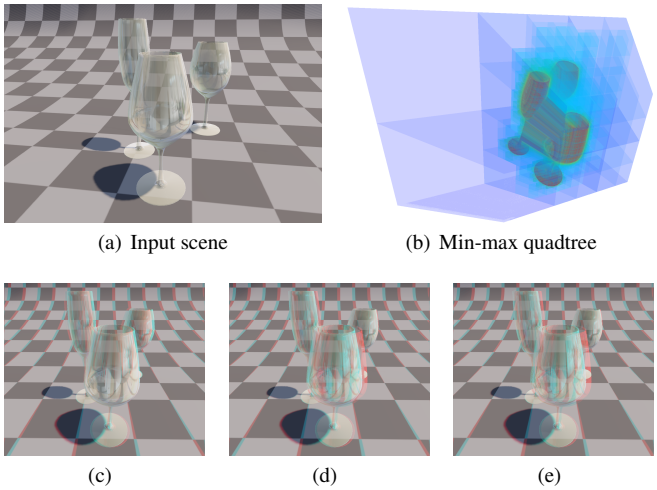
Fig. 5. (a) In the cross-kernel surface reduction, a cell of four pixels is classified as a surface if all eight collinearity tests (marked as red lines) with the adjacent pixels are true. (b) In contrast, the partial cross-kernel surface reduction considers a cell of four pixels as a surface if at least one of the four configurations reports collinearity.

Fig. 6. Merge modes of the irregular grid: There are eight possibilities of merging adjacent cells to form a cell with rectangles.

type is assigned based on the connectedness of its child nodes. All eight possible configurations are shown in Figure 6. Since there are eight different merge types, three additional bits are required to encode the merge type of each node. Based on this information, the grid generation can produce a sparser grid because some adjacent cells will be merged into one rectangular cell. An example quadtree generated by this reduction is depicted in 4(d). With this approach, even fewer primitives are generated.

4.1.4 Building the grid

The reduction strategies are extensions of each other and are only used separately. Each strategy will result in a different quadtree. The surface-estimation quadtree is then used to create an adaptive grid in multiple iterations. Generation starts with a screen-sized grid with the same tile size used for the generation of the surface-estimation quadtree. Thus, each cell of the grid corresponds to a quadtree. Then, in each iteration, the connectedness information and the accumulated estimation error stored per node are used to decide if a grid cell needs to be split. If a node is connected but exceeds a certain estimation error threshold, it is split to avoid geometric distortions during warping. Note that such a split would not affect the connectedness of the grid because the continuity information is stored separately. This process is repeated until the leaf level is reached. The size of the resulting grid cells varies between one pixel and the initial cell size. The adaptive grid generation can be efficiently implemented using a mipmap pyramid for storing the surface-estimation quadtrees and a transform feedback loop for the multi-pass grid refinement.



(a) Input scene (b) Min-max quadtree

(c) (d) (e)

Fig. 7. For a rendered scene (a), a min-max quadtree is used to accelerate the ray casting of semi-transparent fragments (b). The color coding shows the number of cells a ray traverses. The anaglyph 3D images (c) to (e) show the result of three different warping approaches for semi-transparencies. (c) Using conventional image warping based on depth values written only by opaque geometry, the glass appears to be painted on the floor (SSIM=0.885). (d) If semi-transparent geometry contributed to the depth buffer as well, the floor behind the glass seems to be part of the glass texture in the warped stereo image (SSIM=0.957). (e) These artifacts can be avoided using our approach (SSIM=0.975).

4.2 Min-Max Quadtree Generation

Transparent fragments stored in the A-Buffer are re-projected using ray casting. For each pixel in the destination image, a ray needs to be generated and transformed into the source image space. These rays easily reach a considerable length and sampling all pixels along the ray is not feasible in practice. Furthermore, in single-pass A-Buffer implementations (e.g. [11]), the fragment data is typically scattered in memory, which results in incoherent memory access. However, often only a few samples along the ray will actually contribute to the final color. Therefore, a min-max quadtree is used to allow for empty space skipping.

The data structure is generated in two main steps on the GPU. At first, the minimum and the maximum depth of the semi-transparent fragments are gathered per pixel. Then, a series of parallel reductions is performed to create the quadtree bottom-up. On each level, it propagates only the minimum and maximum depth of its four children. The number of required passes depends logarithmically on the resolution of the source image. In order to benefit from texture caching, the quadtree is stored as a mip-map pyramid. The quadtree is recreated every reference frame. However, it is independent of the warp direction. Once it is created, it can be used to perform multiple consecutive warps.

(a) Disocclusions (b) Stretched (c) Our method

Fig. 8. Result of different inpainting hole-filling methods for the oilrig scene.

Fig. 9. This example shows five different levels of the filtered image pyramid. Holes and the foreground object dissolve in higher levels because only non-hole pixels and pixels with a depth larger than the kernel average are taken into account for filtering.

5 RE-RENDERING

Based on the latest input, the view for each eye is re-rendered in three stages. In the first stage, the adaptive warp grid is used to reproject all opaque geometry. In the second stage, the intermediate result is passed to a low-pass filter to mitigate potential dis-occlusion or aliasing artifacts. Finally, a ray is generated for each pixel and intersected with the fragments stored in the A-Buffer to gather and blend all semi-transparent objects.

5.1 Grid Warping

In this stage, the actual 3D warping of opaque pixels is performed by applying the warp function to all vertices of the adaptive warping grid, see Figure 4. In particular, a depth value from the source image is retrieved for each grid vertex. Using this depth and the inverse view projection matrix, all grid vertices are re-projected to world space. Finally, the view projection matrix of the target camera is used to project the grid vertices to the target camera's clip space. Since the projection and view matrices will not change during one warp, the matrix multiplications can be precomputed to one combined warp matrix. Thus, only a single matrix multiplication is required to warp a grid vertex.

However, special attention has to be paid to which depth values are used. Existing grid warping approaches assume vertices between pixels. Consequently, adjacent grid cells are properly stitched together as their corners use the same depth. While this prevents the generation of holes, foreground and background objects will be connected, which is usually not desired as it produces similar artifacts as stretched inpainting, see Figure 8(b). In contrast, placing grid corners directly on pixels would lead to a large number of micro holes since neighboring grid vertices do not use the same depth value.

Instead, we use the connectedness information stored in the surface-estimation quadtree. Grid vertices fetch their depth between pixels, if the adjacent cells form a connected surface, and directly from pixels, if there is a depth discontinuity between the cells. At this point, the connectedness to diagonal neighbors is used as well. The warped grid produces neither rubber sheets nor an excessive number of micro holes.

Another important challenge in grid warping is the texture filtering used. When the adaptive grid has been rasterized in target image space, a texture lookup into the source image has to be performed per fragment. Since this lookup location will not coincide with source pixel locations, interpolation becomes necessary. If linear interpolation is used, good results are achieved on smooth surfaces. At object boundaries, however, color would bleed between foreground and background. This sub-pixel effect would cause serious issues with hole-filling strategies, such as inpainting, because the incorrect color would be extended into the generated holes. On the other hand, nearest-neighbor filtering prevents these issues but causes aliasing artifacts on other surfaces.

Our solution to this issue is similar to the presented grid corner placement. The information generated in the preprocessing stage allows for an adaptive solution: we calculate the appropriate texture coordinates using the surface-estimation quadtree. If the current pixel is in the vicinity of a depth discontinuity (i.e. none of the corresponding collinearity bits is set), nearest neighbor interpolation has to be used. Otherwise, a surface is assumed that requires linear interpolation.

5.2 Hole Filling Using Depth-Based Low-Pass Filter

For hole filling, an epipolar search combined with an adaptive low-pass filtering may increase the visual quality significantly [13]. Mark blurred the generated stripes with increasing distance from the hole's boundary. However, an efficient GPU implementation is far from obvious.

Our solution is to generate multiple low-pass filtered versions of the intermediate warping result, each version being blurred more than the previous, as shown in Figure 9. In particular, the filter uses depth information to avoid foreground objects from occluding valid background information. Before averaging the color of pixels within a certain kernel size, their mean depth is computed. Only pixels which are neither a hole nor closer to the camera than the average depth are taken into account. This guarantees that foreground objects are gradually dissolved by the low-pass filtering. The resulting images are stored in a mip-map pyramid.

Finally, the mip-map pyramid is used to fill holes caused by dis-occlusions. The lookup level in the pyramid depends on the distance to the background border of the hole. This border is searched in epipolar direction with an exponentially increasing step size. This is possible because the filter radius is effectively doubled in each layer of the mip-map pyramid. This yields a very conservative hole border distance estimate because a precise nearest-neighbor search would be too slow.

5.3 Ray Casting Transparencies in the A-Buffer

The ray casting algorithm performs on the min-max quadtree, see Section 4.2, and the A-Buffer. The traversal of the min-max quadtree is inspired by the stackless height-field ray casting algorithm presented by Tevs et al. [27]. The generated ray enters the min-max quadtree at its root node. Its exit intersection with the root node is computed in screen space using the formulas presented by Dick et al. [5]. The quadtree is traversed until an intersection with a leaf node occurs. Compared to the original traversal, the algorithm cannot stop, if an intersection is found. Instead it continues to gather transparent fragments along the ray until either the accumulated opacity exceeds a given threshold or the ray reaches the depth of the warped grid.

Thus far, a leaf node can be found efficiently; however, the possibility to move upwards in the quadtree structure is required in order to find secondary hits or to regain traversal speed when a ray travels close to a semi-transparent surface without actually hitting it. We follow the approach of Tevs et al. to ascend one level whenever the ray start advances to a boundary which is also a boundary between nodes one level above. That means, whenever a node has no sibling in the ray direction, the traversal will ascend one level.

At leaf nodes, the corresponding linked lists from the A-Buffer are searched for intersections and the retrieved colors along the ray are composited. In practice, intersecting these fragments would cause aliasing artifacts because adjacent fragments do not form a water-tight surface. A ray could pass between them which would result in micro holes. Therefore, we extrude each fragment slightly in depth and intersect the resulting box. We make sure that the ray does not hit any other fragment within the thickness of the box to avoid multiple intersections with the same surface.

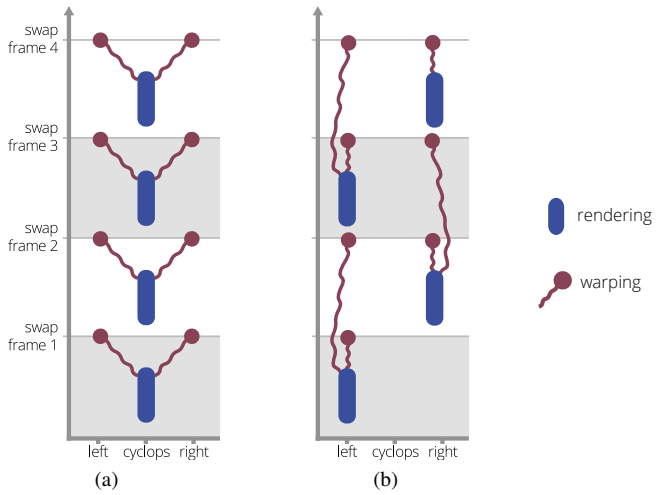


Fig. 10. Our two proposed image warping strategies. Cyclops warping from a central perspective (a) and alternate frame warping for two consecutive frames (b).

6 Evaluation

As mentioned in Section 3, the presented algorithms were integrated into an open-source rendering engine [Schollmeyer2015]. This framework implements programmable transparency and is therefore well-suited for a test implementation. The underlying deferred rendering pipeline was extended with two different warping strategies, see Figure 10. *Cyclops warping* warps a frame from a central rendered perspective into the left and right eye. *Alternate frame warping* exploits temporal coherence by warping the left and right reference views for two consecutive frames. Each strategy can be enabled instead of conventional stereo rendering. Depending on the type of interaction and 3D content, both approaches have their advantages and disadvantages which will be discussed later.

All tests were performed on a 3.33 GHz Intel Core i7 workstation with 12GiB RAM equipped with a single NVIDIA GeForce GTX 980 GPU with 4GiB video memory. All performance timings were measured for a rendering resolution of 1920x1080, if not stated otherwise. The maximum node size for both the surface estimation quadtree and the min-max quadtree was set to 32x32 pixel. In all tests, except the comparison of the different surface estimation strategies, the irregular grid reduction was used for the adaptive grid generation. The test scenes are depicted in Figure 11.

6.1 Results

A detailed comparison of the timings for the adaptive grid generation and warping is shown in Figure 12. For all scenes, the proposed reduction strategies perform much better than the method proposed by [6] because they produce far fewer primitives. Mostly, the irregular grid strategy will produce the smallest number of grid cells. However, a significant difference between our strategies can only be observed for the extreme hairball scene.

Furthermore, we evaluated the costs for the different warping stages. The graph, shown in Figure 14, indicates that the performance of most stages is hardly affected by the scene complexity. Only for the hairball, the grid generation does not benefit from the adaptive reduction which results in more primitives that need to be reprojected, as shown in Figure 13.

We also evaluated the image quality of our adaptive grid warping. Table 1 shows a comparison of the resulting image errors between our method and an implementation of [6]. For better comparability, both methods used our hole-filling approach. The results indicate that although less triangles are used for warping, the image quality is almost equal. In two cases, our results are even slightly closer to ground truth. This is because of two reasons. The adaptive grid generation prevents an over-tessellation of slanted surfaces which may cause aliasing artifacts after reprojection. Furthermore, the adaptive texture filtering at depth discontinuities improves the quality of the hole-filling. In conclusion, our method delivers the same quality as Didyk et al. while being significantly faster (see Fig. 12) in most cases.

Table 1. This table shows the image warping error for our test scenes. For all views, the image quality of our approach is almost equal to Didyk et al. In some cases, it is even slightly better due to the adaptive texture filtering and the prevention of over-tessellation.

	Error metric	Dydik et al.	Our method
Oilrig	PSNR	**25.4221**	25.4061
	SSIM	**0.95673**	0.95647
Hairball	PSNR	20.6883	**20.7276**
	SSIM	0.90727	**0.90774**
Sponza	PSNR	35.6339	**35.8308**
	SSIM	0.99141	**0.99148**

In addition, we evaluated how our algorithms scale with increasing image resolutions. Figure 13 shows that the number of primitives in the grid increases sublinearly with higher resolutions due to the adaptive grid generation. The corresponding timings of the different stages are illustrated in Figure 15. Ray casting and hole filling can be efficiently performed in a single pass as they both operate on a per-pixel level. Therefore, the corresponding timings appear as one stage in the graph. Although, some stages exhibit sublinear behavior, most stages also include parts that linearly depend on the image resolution, e.g. the rasterization of the grid and the ray casting.

As shown in Figure 7, ray casting the A-Buffer for warping translucent fragments results in images hardly distinguishable from the ground truth image. This was also confirmed by the corresponding image error metrics. The ray casting stage requires less than 1ms for the Sponza scene with the translucent window gallery, as shown in Figure 14. In general, the performance may depend on the application scenario. It is interesting to observe that an increasing warp disparity affects the performance of the ray casting much more than a high depth complexity of semi-transparent fragments. If per-fragment programmable transparency is required, ray casting of the A-Buffer is probably the only suitable option because an efficient forward warping algorithm for these fragments has not been shown yet. For scenes with constant per-object opacity values, however, separate rendering and compositing of semi-transparent objects may be in some cases more efficient than our approach.

We also evaluated our hole-filling method in comparison to other approaches. Therefore, we compared warped images with ground truth images by using various image quality metrics. In the related literature, other researchers used the peak signal-to-noise ratio (PSNR) and the structural similarity index (SSIM) [29]. The latter was proven to correlate better with perceived quality than PSNR [9]. However, we will present both for better comparability. The results are shown in Table 2. For the given test scenes, our method provides the highest quality, but was only about 0.3ms slower compared to stretched inpainting. The kernel-size of the filter and the reduction heuristic both allow for tradeoffs between quality and performance.

Finally, we briefly investigated the potential gain in visual quality when combining our approach with an output-sensitive rendering system. For this purpose, we integrated a renderer for level-of-detail point clouds, similar to [8]. In this system, the performance can be increased by decreasing the model fidelity. However, if image warping was enabled, a higher level-of-detail could be rendered at the same frame rates. We assumed that the resulting increase in image quality would outweight potential warping artifacts. Figure 1 shows some results for a rock scene which consists of about 400 million points at the highest detail. For the view shown in 1(a), a pixel-accurate stereoscopic ren-

(a) Oilrig (b) Hairball (c) Sponza

Fig. 11. This Figure shows our test scenes with different complexities: The oilrig (a) has some large surfaces, but also many small geometries and occlusions. The hairball (b) represents a worst case scenario for the grid generation as it has a high frequency of depth discontinuities. For the evaluation of the ray casting, the standard Sponza scene (c) was extended with galleries of semi-transparent windows.

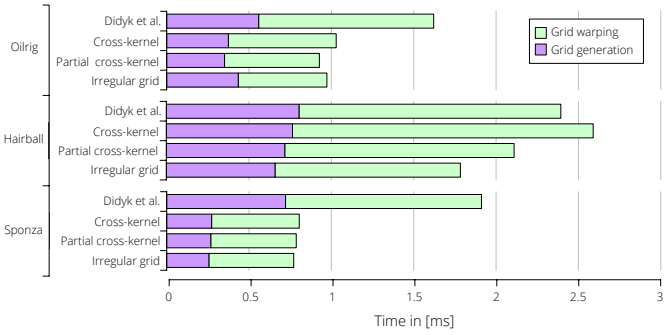


Fig. 12. Comparison of different surface estimation strategies.

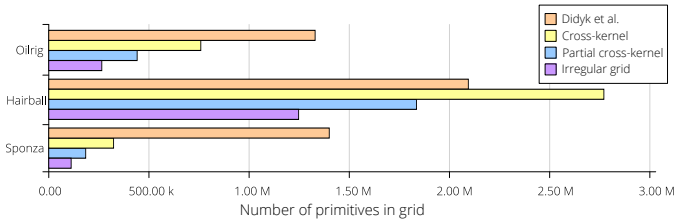


Fig. 13. Comparison of number of primitives in the adaptive grid for different surface estimation strategies.

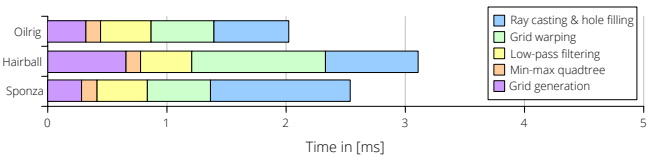


Fig. 14. These timings show that all stages perform almost scene-independently. In the extreme hairball scene, grid generation and grid warping stages are slower because they do not benefit equally from the adaptive grid generation. For sponza, the costs for ray casting are slightly higher because semi-transparent surfaces are visible.

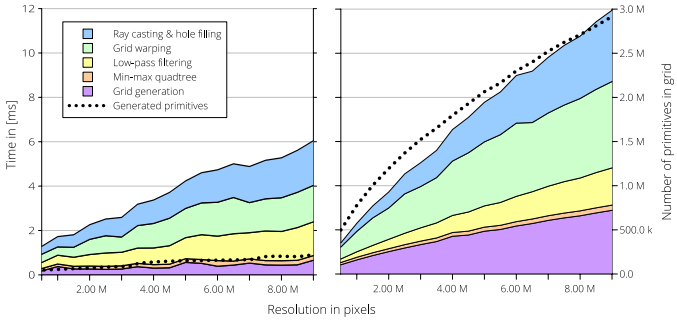


Fig. 15. This graph shows the contributions of the different stages to the warping time for different resolutions of the oilrig (left) and the hairball scene (right).

dering performs at about 20Hz. A corresponding close-up is shown in Figures 1(b) to 1(d). If a frame rate of 60Hz was required, the necessary decrease of fidelity led to a perceivable loss of detailed features (SSIM=0.945), as shown in 1(c). In contrast, using image warping, the resulting image quality loss could hardly be seen (SSIM=0.990).

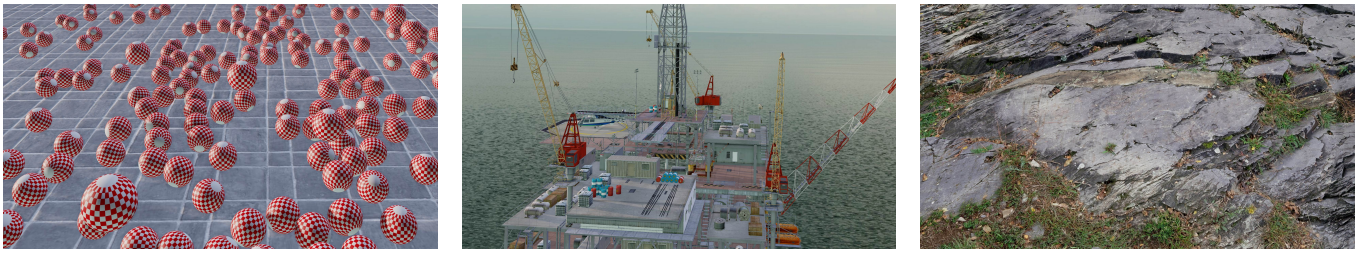
6.2 User study

The presented algorithms can increase the efficiency, quality and applicability of stereoscopic rendering via image warping. However, image warping may introduce visual artifacts which can hardly be evaluated using quantitative error metrics. Therefore, a user study was conducted with 16 participants (2 female, 14 males) between 20 and 38 years old, each experienced in 3D computer graphics and the usage of stereoscopic displays.

6.2.1 Study Design

The users were head-tracked and they could freely move in front of a 27-inch passive stereoscopic display with a resolution of 2560x1440 pixels for each eye. As shown in Figure 17(a), the display employs a semi-transparent mirror and two monitors with Nvidia G-Sync support. We chose this display because we did not get access to the raw position and orientation tracking data of recent HMDs. Furthermore, the performance gained by image warping directly results in higher frame rates without screen tearing because of the adaptive synchronization between graphics hardware and monitor via G-Sync. The evaluation was performed using conventional rendering and two warping strategies which are illustrated in Figure 10. The users were asked to rank the different rendering modes by their preference. They were allowed to indicate ties because a forced choice may have distorted the result. Furthermore, the users answered detailed Likert-scale questions about the perceived pleasantness, performance, image quality and latency while they were using the system.

Our study was focused on the evaluation of the grid warping and the hole-filling strategies. There were three different test scenes: the rock (Figure 16(c)), the oil rig (Figure 16(b)) and a scene with many textured balls (Figure 16(a)) which were falling on the ground, colliding and rolling away. The ball scene was chosen to investigate how image warping is perceived in highly dynamic scenes. In contrast, the rock and the oil rig scenes are both static scenes and were used to investigate how image warping is perceived for highly detailed scenes. However, while the oil rig has a high depth complexity with many potential un-occlusions, the rock scene is highly realistic with many little features on the rock. Figure 17(b) shows the average frame rates for all scenes and rendering modes. The oil rig scene rendered at about 43Hz with

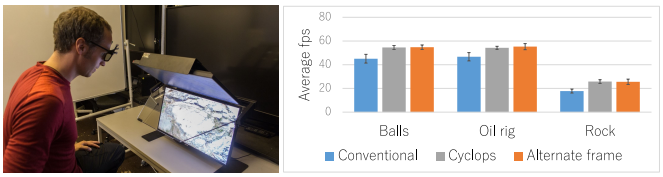


(a) Balls (b) Oil rig (c) Rock

Fig. 16. In the user study, the participants were shown three different scenes: a physics simulation showing many bouncing balls (a), the oil rig (b) and a highly-detailed rock scene (c). The frame rates for the test scenes in all rendering modes are shown in Figure 17(b).

Table 2. Image quality results for various hole-filling strategies: For each result, the luminance of the warped image and of a ground-truth image were compared. **Green** numbers are the best results for each metric while **red** numbers are the worst results.

	Error metric	Rubber sheets	Stretch inpaint	Our method
Oilrig	PSNR	**24.36**	24.67	**25.91**
	SSIM	**0.92580**	0.94359	**0.95277**
Hairball	PSNR	20.60	**20.42**	**21.60**
	SSIM	0.87819	**0.87642**	**0.90650**
Sponza	PSNR	**35.45**	37.41	**37.63**
	SSIM	**0.99449**	**0.99486**	0.99461



(a) Display setup (b) Performance

Fig. 17. The rendering performance was logged while participants moved in front of the stereoscopic display (a). The average frame rates and their standard deviation are shown in (b).

regular stereoscopic rendering and at about 55Hz with warping, the rock with 18Hz and 26Hz, and the balls scene with 45Hz and 55Hz. The regular end-to-end latency of our system depends on the frame rate and ranges from about 70ms at 60Hz to about 105ms at 20Hz and was measured based on Friston et al. [7]. Furthermore, we limited user interaction to tracked head-movements to avoid confounding influences of navigation parameters, viewing angles and potentially varying frame rates.

Two separate tests were performed. In both tests, the scenes were shown in random order. In the first test, for each scene the participants could switch between three render modes in shuffled order: conventional stereoscopic rendering, cyclops warping and alternate frame warping. In the second test, each scene was rendered using alternate frame warping and the users could choose between our hole-filling method and stretching.

6.2.2 Study Results

The results of our study are shown in Figure 18. They indicate that the participants' general preferences depend on the scene content. For the rock scene, both warping approaches were preferred to conventional rendering. For the oil rig, conventional rendering and alternate frame warping were preferred, while the latter was quite unpopular for

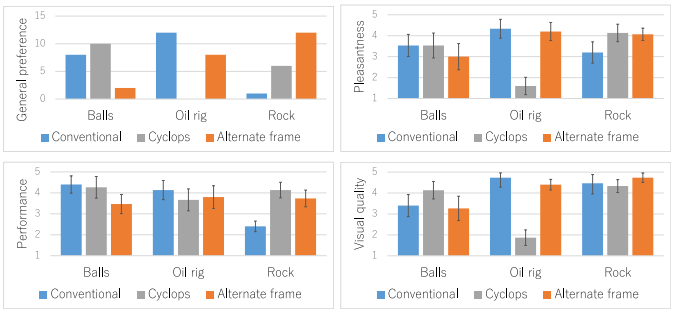


Fig. 18. These are the results for the warping strategy evaluation. The upper left diagram shows the preferred modes for all participants. If a user indicated a tie between two preferred modes, both were counted. The other diagrams show the mean values and standard deviations of the answers to the detailed questions about the perceived pleasantness, performance and visual quality.

rendering the ball scene.

The answers to our detailed questions point to the reasons for these results. The participants replied to these questions on a 5-point Likert scale ranging from worst (1) to best (5). For the evaluation of the answers, we used two-sided, paired sample t-tests and an α-value of 0.05. After the Bonferroni adjustment, we obtain statistical significance for t-values greater than 7.6488 or p-values lower than 0.0166 respectively.

For all three scenes, the general preferences mostly coincide with the perceived pleasantness. In particular, for the rock scene alternate frame warping ($t(2) = 19.8024, p = 0.0025$) and cyclops warping ($t(2) = 9.8042, p = 0.01024$) were considered significantly more pleasant than conventional rendering, mainly because both were perceived more fluent ($t(2) = 5011.1, p = 3.9810^{-8}$ for alternate frame warping and $t(2) = 51.178, p = 0.0004$ for cyclops warping) while no difference in visual quality was observed. In contrast, many visual artifacts were noticed for the cyclops warping of the oil rig ($p \approx 0$) because of its high depth complexity and the resulting un-occlusions.

For the ball scene, users found alternate frame warping quite unpleasant compared to conventional rendering ($t(2) = 6.8375, p = 0.0207$). The lower update rate of dynamic objects for the alternate frame warping and the conventional rendering was interpreted as visual artifacts by many users. Surprisingly, the image errors of cyclops warping were barely noticed for this scene.

The outcome of the second test, in which we compared the different hole-filling approaches, did not show significant differences. Preliminary tests on a stereoscopic, projection-based display indicated that differences are perceivable on larger screens. Further studies are necessary, but remain future work.

In conclusion, the study confirms that image warping was preferred to conventional rendering for two of three scenes and visual artifacts went mostly unnoticed. However, the choice of the warping strategy highly depends on the dynamics in the scene.

6.3 Discussion and Limitations

In this work, we focused on the development of scalable image-warping algorithms for high frame-rate, low-latency stereoscopic rendering systems. We suggested two warping strategies: cyclops warping and alternate frame warping. Cyclops warping updates the positions of moving objects in each frame whereas alternate frame rendering has the disadvantage that it can only update moving objects every second frame. However, the beauty of alternate frame warping is that it converges to correct and artifact-free images when the user slows down. Furthermore, additional per-pixel motion information could be used to improve the warping of dynamic objects [25, 26] for both warping approaches, but in particular for the alternate frame warping. Although, this could be added easily for the grid-based forward warping of opaque pixels, an adaptation of our ray-casting based warping of translucent fragments is quite intricate. In particular, it would require to reinsert all dynamic fragments into the A-Buffer at their new positions and thereby also trigger a recreation of the corresponding min-max quadtree. For dynamic light sources, per-pixel object ids could be used to reshade the warped fragments. However, in most modern rendering engines the shading computations are quite costly. If they would be performed for each pixel in the target image, grid warping may become a bottleneck for high resolutions.

The results of the warping process can be further improved by adding prediction of head movements which is currently not supported in our system. Prediction could not only minimize the perceived latency; it would also decrease the warping distance and therefore improve the visual image quality of the warp. Our approaches can be also implemented if the rendering of reference frames and warping are performed asynchronously [25]. However, for an implementation on a single graphics card effective fine-granular preemption and a high-priority context are needed. Both are not yet available in OpenGL.

The applicability of our warping approach depends strongly on the achievable frame rates for rendering the scene, the image resolution and the complexity of the scene. Figure 15 shows that for today's image resolutions of head-mounted displays of around two to three megapixels for both eyes together (a single display stretching across both eyes is typically used), the complete warping for the left and right eye can be performed in two to four milliseconds on current GPUs. If we want to achieve a frame rate of 90Hz for stereoscopic applications, this leaves us with only 5.5ms for rendering each eye. If we use warping instead, and assume a cost of 3ms, we can spend 8ms on rendering one view. That corresponds to a 45% increase in geometry, shading complexity or generally visual quality. At the same time the latency is reduced by 8ms (11ms-3ms). This relationship improves for lower frame rates, e.g. at 60Hz we have about 8ms per eye for regular stereoscopic rendering. We can spend 13ms on rendering if we use 3ms on warping both eyes. That corresponds to more than a 60% increase in rendering time for the scene. At the same time, the latency decreases by about 13ms. This confirms that the technique is a good match for today's GPU and HMD configurations. Figure 15 also reveals that warping would not be worthwhile on current GPUs if the resolution of the head-mounted displays increases to 4 or even 8 megapixels. However, we expect that the performance of the GPUs also increases in lockstep and thus our techniques will remain useful in the future.

The memory requirements of the presented algorithms are quite decent. For full-HD resolution, all required intermediate data structures sum up to 68.6MiB. However, the implementation of an A-Buffer involves the allocation of a fixed memory budget which may require hundreds of megabytes depending on the image resolution and the application scenario.

7 Conclusion and Future Work

Our algorithms increase the scalability and applicability of image warping as a means to improve the frame rate and latency of stereoscopic rendering systems. We presented three novel reduction schemes to generate a surface-estimation quadtree for a depth image. Based on the continuity information stored in this data structure, an adaptive grid for positional image warping is generated. We have shown that our approach yields fewer primitives than other methods and therefore decreases the warping overhead. An important feature of our algorithm is that the detected continuity is used to adaptively adjust the grid vertices' position and color texture lookup such that both aliasing and stretching artifacts are mostly prevented and become almost imperceptible in head-tracked environments. Furthermore, we suggest an efficient method for ray casting translucent fragments stored in an A-Buffer, which integrates well into our warping. Potential artifacts due to disocclusions are mitigated by a depth-based low-pass filter.

In order to confirm these claims, we integrated two warping strategies based on our algorithms into an open-source rendering engine and performed quantitative tests considering the image quality and performance as well as a user study. Performance improvements are scene-dependent and are typically in the range of 20 to 40% in our scenarios. Image quality for stereoscopic warping was not significantly decreased compared to conventional rendering and was hardly noticed by users in head-tracked environments. The evaluation of our user study reveals that rendering modes with image warping were preferred for two test scenarios.

Ray casting the A-Buffer has far fewer disocclusion artifacts than the forward warping of the depth buffer since the A-Buffer effectively contains a rasterised version of the complete set of (partially) visible translucent surfaces. Thus re-rendering the discretized scene description contained in the A-Buffer from a slightly different perspective works well. It also performs well if the number of visible translucent fragments is limited. However, this approach cannot be easily transferred to larger scenes of opaque objects with non-trivial depth complexity since it becomes inefficient to store a discretized version of the entire scene in a multi-layered G-Buffer even though this would have the advantage that G-Buffer and A-Buffer could be merged into one buffer. Nevertheless, we think that there is potential to apply this idea to a conservative approximation of the potentially visible set of opaque and transparent objects instead of the entire scene.

Our efficient and high-quality image warping contributes to the arsenal of practical solutions to respond to the challenges of modern virtual reality applications such as stereoscopic rendering, effective handling of transparencies, high frame rates and low latency.

Acknowledgments

This work was supported by the German Federal Ministry of Education and Research (BMBF) under grant 03IPT704X (project Big Data Analytics). We would also like to thank Sven Bertel and Stefanie Wetzel from the Usability Research Group at Bauhaus-Universität Weimar for their support on the statistical evaluation of the user study.

References

[1] S. J. Adelson and L. F. Hodges. Generating exact ray-traced animation frames by reprojection. *Computer Graphics and Applications*, 15(3):43–52, May 1995. doi: 10.1109/38.376612

[2] H. Bowles, K. Mitchell, R. W. Sumner, J. Moore, and M. Gross. Iterative image warping. *Computer Graphics Forum*, 31(2pt1):237–246, 2012. doi: 10.1111/j.1467-8659.2012.03002.x

[3] L. Carpenter. The a -buffer, an antialiased hidden surface method. In *Proceedings of the 11th Annual Conference on Computer Graphics and Interactive Techniques*, SIGGRAPH '84, pp. 103–108. ACM, New York, NY, USA, 1984. doi: 10.1145/800031.808585

[4] S. E. Chen and L. Williams. View interpolation for image synthesis. In *Proceedings of the 20th Annual Conference on Computer Graphics and Interactive Techniques*, SIGGRAPH '93, pp. 279–288. ACM, 1993. doi: 10.1145/166117.166153

[5] C. Dick, J. Krüger, and R. Westermann. Gpu ray-casting for scalable terrain rendering. In *Proceedings of Eurographics 2009 - Areas Papers*, pp. 43–50, 2009.

[6] P. Didyk, T. Ritschel, E. Eisemann, K. Myszkowski, and H.-P. Seidel. Adaptive image-space stereo view synthesis. In *Vision, Modeling and Visualization Workshop*, pp. 299–306. Siegen, Germany, 2010.

[7] S. Friston and A. Steed. Measuring latency in virtual environments. *IEEE Transactions on Visualization and Computer Graphics*, 20(4):616–625, April 2014. doi: 10.1109/TVCG.2014.30

[8] P. Goswami, Y. Zhang, R. Pajarola, and E. Gobbetti. High quality interactive rendering of massive point models using multi-way kd-trees. In *18th*

Pacific Conference on Computer Graphics and Applications, pp. 93–100, 2010.

[9] P. Hanhart and T. Ebrahimi. Quality assessment of a stereo pair formed from decoded and synthesized views using objective metrics. In *3DTV-Conference: The True Vision-Capture, Transmission and Display of 3D Video*, pp. 1–4. IEEE, 2012.

[10] S.-F. Hsiao, J.-W. Cheng, W.-L. Wang, and G.-F. Yeh. Low latency design of depth-image-based rendering using hybrid warping and hole-filling. In *International Symposium on Circuits and Systems*, pp. 608–611. IEEE, 2012.

[11] S. Lefebvre, S. Hornus, and A. Lasram. Per-Pixel Lists for Single Pass A-Buffer. In W. Engel, ed., *GPU Pro 5: Advanced Rendering Techniques*, pp. 3–23. CRC Press, 2014.

[12] G. Lochmann, B. Reinert, T. Ritschel, S. Müller, and H. Seidel. Real-time reflective and refractive novel-view synthesis. In *VMV 2014: Vision, Modeling & Visualization, Darmstadt, Germany, 2014. Proceedings*, pp. 9–16, 2014. doi: 10.2312/vmv.20141270

[13] W. R. Mark. Post-rendering 3d image warping: Visibility, reconstruction, and performance for depth-image warping. Technical Report TR99-022, 1999.

[14] W. R. Mark, L. McMillan, and G. Bishop. Post-rendering 3d warping. In *Proceedings of the 1997 Symposium on Interactive 3D Graphics*, pp. 7–16. ACM, 1997.

[15] L. McMillan and G. Bishop. Head-tracked stereoscopic display using image warping. In *PROCEEDINGS SPIE, VOLUME 2409*, pp. 21–30, 1995.

[16] D. Nehab, P. V. Sander, J. Lawrence, N. Tatarchuk, and J. R. Isidoro. Accelerating real-time shading with reverse reprojection caching. In *Proceedings of the 22Nd ACM SIGGRAPH/EUROGRAPHICS Symposium on Graphics Hardware*, GH '07, pp. 25–35. Eurographics Association, Aire-la-Ville, Switzerland, Switzerland, 2007.

[17] M. M. Oliveira, B. Bowen, R. McKenna, and Y. Chang. Fast digital image inpainting. In *Proceedings of the IASTED International Conference on Visualization, Imaging and Image Processing (VIIP 2001), Marbella, Spain, September 3-5, 2001*, pp. 261–266, 2001.

[18] E. M. Peek, B. C. Wünsche, and C. Lutteroth. Image warping for enhancing consumer applications of head-mounted displays. In *Proceedings of the Fifteenth Australasian User Interface Conference-Volume 150*, pp. 47–55. Australian Computer Society, Inc., 2014.

[19] N. Plath, L. Goldmann, A. Nitsch, S. Knorr, and T. Sikora. Line-preserving hole-filling for 2d-to-3d conversion. In *Proceedings of the 11th European Conference on Visual Media Production*, pp. 8:1–8:10. ACM, 2014. doi: 10.1145/2668904.2668931

[20] N. Plath, S. Knorr, L. Goldmann, and T. Sikora. Adaptive image warping for hole prevention in 3d view synthesis. *IEEE Transactions on Image Processing*, 22(9):3420–3432, Sept 2013. doi: 10.1109/TIP.2013.2268940

[21] T. Saito and T. Takahashi. Comprehensible rendering of 3-d shapes. In *Proceedings of the 17th Annual Conference on Computer Graphics and Interactive Techniques*, SIGGRAPH '90, pp. 197–206. ACM, New York, NY, USA, 1990. doi: 10.1145/97879.97901

[22] D. Scherzer, L. Yang, O. Mattausch, D. Nehab, P. V. Sander, M. Wimmer, and E. Eisemann. Temporal coherence methods in real-time rendering. *Computer Graphics Forum*, 31(8):2378–2408, 2012. doi: 10.1111/j.1467-8659.2012.03075.x

[23] A. Schollmeyer, A. Babanin, and B. Froehlich. Order-independent transparency for programmable deferred shading pipelines. *Computer Graphics Forum*, 34(7):67–76, 2015. doi: 10.1111/cgf.12746

[24] J. Shade, S. Gortler, L.-w. He, and R. Szeliski. Layered depth images. In *Proceedings of the 25th Annual Conference on Computer Graphics and Interactive Techniques*, SIGGRAPH '98, pp. 231–242. ACM, New York, NY, USA, 1998. doi: 10.1145/280814.280882

[25] F. A. Smit, R. van Liere, S. Beck, and B. Fröhlich. An image-warping architecture for vr: Low latency versus image quality. In *Virtual Reality Conference, 2009. VR 2009. IEEE*, pp. 27–34, March 2009. doi: 10.1109/VR.2009.4810995

[26] F. A. Smit, R. van Liere, and B. Fröhlich. The design and implementation of a vr-architecture for smooth motion. In *Proceedings of the 2007 ACM Symposium on Virtual Reality Software and Technology*, VRST '07, pp. 153–156. ACM, New York, NY, USA, 2007. doi: 10.1145/1315184.1315212

[27] A. Tevs, I. Ihrke, and H.-P. Seidel. Maximum mipmaps for fast, accurate, and scalable dynamic height field rendering. In *Proceedings of the 2008 symposium on Interactive 3D graphics and games*, pp. 183–190. ACM, 2008.

[28] J. Viega, M. J. Conway, G. Williams, and R. Pausch. 3d magic lenses. In *Proceedings of the 9th Annual ACM Symposium on User Interface Software and Technology*, pp. 51–58. ACM, 1996. doi: 10.1145/237091.237098

[29] Z. Wang, A. C. Bovik, H. R. Sheikh, and E. P. Simoncelli. Image quality assessment: From error visibility to structural similarity. *IEEE Transactions on Image Processing*, 13(4):600–612, Apr. 2004. doi: 10.1109/TIP.2003.819861

[30] L. Westover. Footprint evaluation for volume rendering. In *Proceedings of the 17th Annual Conference on Computer Graphics and Interactive Techniques*, SIGGRAPH '90, pp. 367–376. ACM, New York, NY, USA, 1990. doi: 10.1145/97879.97919

[31] L. Yang, Y.-C. Tse, P. V. Sander, J. Lawrence, D. Nehab, H. Hoppe, and C. L. Wilkins. Image-based bidirectional scene reprojection. In *Proceedings of the 2011 SIGGRAPH Asia Conference*, SA '11, pp. 150:1–150:10. ACM, New York, NY, USA, 2011. doi: 10.1145/2024156.2024184

[32] L. Zhang and W. J. Tam. Stereoscopic image generation based on depth images for 3d tv. *IEEE Transactions on Broadcasting*, 51(2):191–199, June 2005. doi: 10.1109/TBC.2005.846190