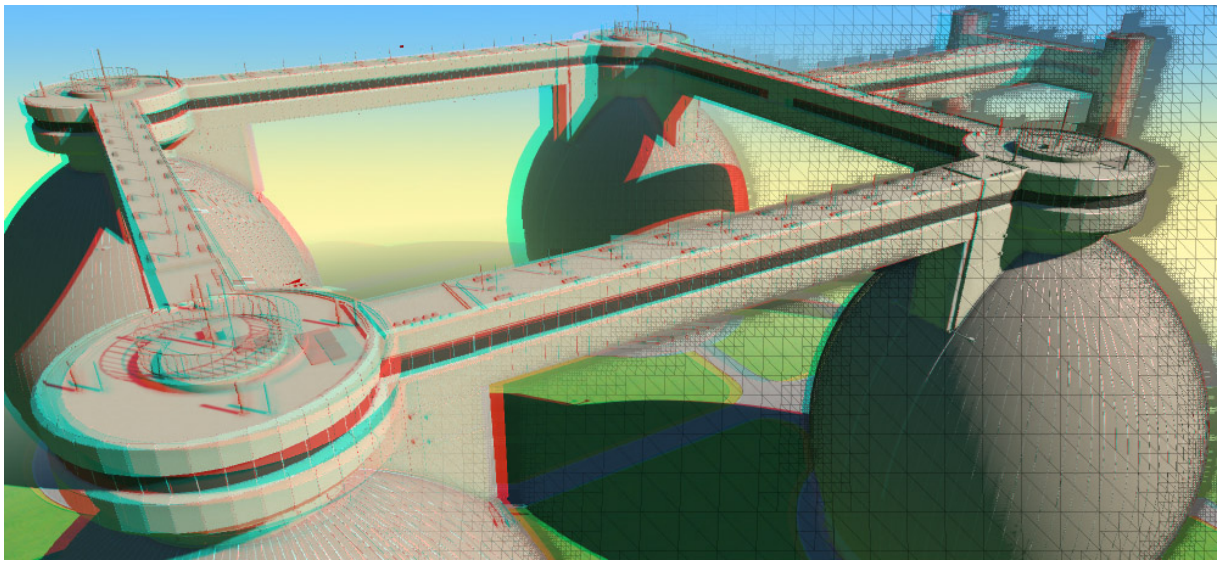
Vision, Modeling, and Visualization (2010)
Reinhard Koch, Andreas Kolb, Christof Rezk-Salama (Eds.)

Adaptive Image-space Stereo View Synthesis

Piotr Didyk[1] Tobias Ritschel[2,3] Elmar Eisemann[2,3] Karol Myszkowski[1] Hans-Peter Seidel[1]

[1]MPI Informatik, Saarbrücken [2]Télécom ParisTech / CNRS, Paris [3]Intel Visual Computing Lab, Saarbrücken



Abstract

Stereo vision is becoming increasingly popular in feature films, visualization and interactive applications such as computer games. However, computation costs are doubled when rendering an individual image for each eye. In this work, we propose to only render a single image, together with a depth buffer and use image-based techniques to generate two individual images for the left and right eye. The resulting method computes a high-quality stereo pair for roughly half the cost of the traditional methods. We achieve this result via an adaptive-grid warping that also involves information from previous frames to avoid artifacts.

1. Introduction

Recently, stereo vision has received much attention due to its high success in feature films, visualization and interactive applications such as computer games. However, 3D vision does not come for free and often implies that two images need to be rendered instead of a single one, as for standard rendering. This can have a high impact on performance which is an issue for real-time applications. In this work, we propose to create only a single view of the scene, together with its depth buffer and use image-based techniques to generate two individual images for the left and the right eye. The resulting stereo effect is of a high quality, but our approach avoids the cost of rendering two individual frames. In this context, we address two major challenges. First, our stereo view-synthesis should show a performance behavior that approaches the rendering time for a single view. Second, as-few-as-possible artifacts should be introduced into the stereo image pair. Our solution addresses both issues via an adaptive algorithm that respects depth disparity, exploits temporal and spatial consistency, and maps well to the GPU.

This paper is structured as follows: After reviewing previous work in Section 2, we propose our algorithm in Sec-

tion 3, for which results are presented in Section 4. Strengths and limitations are discussed in Section 5, before we conclude in Section 6.

2. Previous Work

In this section, we review previous work, which addressed the synthesis of stereo images as well as the perception of stereo.

Depth perception The effect of stereopsis of image pairs is known since the mid-nineteenth century [Whe38]. It was also noted early, how 3D computer graphics is an excellent means to generate stereo images [Mor76], simply by rendering two individual views: one for each eye. The main drawback of such a stereo-view creation is that rendering time is also doubled. Nonetheless, this naïve approach will serve us as a reference to compare to when evaluating our *stereo view synthesis* approach that relies on a single rendered view.

Stereo View-synthesis A surprisingly simple form of stereo view synthesis was proposed as early as 1974 by Ross [Ros74]. Assuming a horizontal moving camera, previous frames look similar to the one eye's view and future frames look similar to the other eye's view. Therefore, playing a video stream with different delays for left and right eyes gives a stereo impression. This approach however is limited to horizontal movements and requires knowledge of the future or introduces delay, which is both unwanted for interaction in virtual worlds. Still, the observation that a previous rendering for one eye can serve as a source for the other eye's view serves as an inspiration to our approach.

Warping Deforming one image into another one is called *warping*, a technique with many applications, as detailed in Wolberg's survey [Wol98]. Warping can be used to synthesize new views [CW93] such as stereo image pairs from single [KKS08] or multiple [ZKU*04] images, in-between frames for video streams of low refresh rates [SLW*08, MHM*09, DER*10].

Image stabilization [LGJA09] makes use of similar strategies, but such optimized deformations are not suitable for stereo upsampling because they would alter the depth disparity for both views, changing the overall stereo impression.

Many methods, such as Stich et al. [SLW*08] and Mahajan et al. [MHM*09], are offline approaches and rely on future frames. In stereo, we cannot exploit such information (it would correspond to "more-left" and "more-right" images to synthesize new views). For interactive applications, we further want the synthesis to run at high refresh rates in the order of just a few milliseconds to improve significantly upon rendering a second view.

The method of Didyk et al. [DER*10] can compute such novel views without knowledge of future frames by exploiting information directly extracted from the 3D scene, where data such as depth is a by-product of the current rendering process. Their original work targets synthesis in time, while we extend their framework to synthesize stereo image pairs.

Knorr et al.'s [KKS08] approach uses structure from motion [HZ00] to generate a dense depth map from a video, which is then used to synthesize new views by reprojection. A strength of their method is, that it can work for images without depth information if they exhibit sufficient features. No performance data is given, but sorting and projecting incoherent, individual points remounts to data scatter which has lower quality and is inferior to gathering when used for warping [DER*10].

Usually pixel projection found application in upsampling schemes for 3D interactive applications. Mark et al. [MMB97] re-use of shaded samples based on depth buffer information. Such approaches are also the basis of the Render Cache [WDP99] which is effective, e. g., in global illumination where samples are very expensive. More recently Nehab et al. [NSL*07] proposed a new caching scheme, exchanging forward for reverse mapping.

Some methods only investigate reduced resolutions [SGH*01], and such a reduction is particularly interesting if subsets of pixels are used to render subsets of views. This fits well to ray-tracing and volume-rendering [DEF*07]. In our method we avoid such interleaving which typically reduces the potential resolution of the output. Our stereo disparity is approximated by warping full resolution frames. Hereby, we preserve high frequencies such as in textures.

Recently, Zhang and co-workers [ZHQ*07] even avoid the construction of a depth map altogether and produce a stereo image by recasting it as an optimization of the parallax effect. Such optimizations are computationally intensive, and are applied to many (also future) frames.

3. Our Approach

In this section, we propose a pipeline (cf. Sec. 3.2) to turn a rendered image with depth into a stereo image pair as shown in the teaser. To this end, we construct a disparity mapping (cf. Sec. 3.1) from an image location in one eye to the image location of the other eye. We observe that this mapping is piecewise smooth, and exploit this fact to efficiently create a high-quality stereo image pair using an adaptive approach (cf. Sec. 3.3). Finally, we discuss how to improve the result further by warping not only between the left and right eye, but also between the current and previous frames (cf. Sec. 3.4). In particular, this modification also ensures convergence (cf. Sec. 3.5) to the reference in the case of a static scenes and a decelerating camera.

3.1. Disparity

Let $\mathbf{y} \in \mathbb{R}^3$ be a point in world space and $\mathbf{x}_{\text{left}} \in \mathbb{R}^2$ its projection into the left eye's view as well as $\mathbf{x}_{\text{right}} \in \mathbb{R}^2$

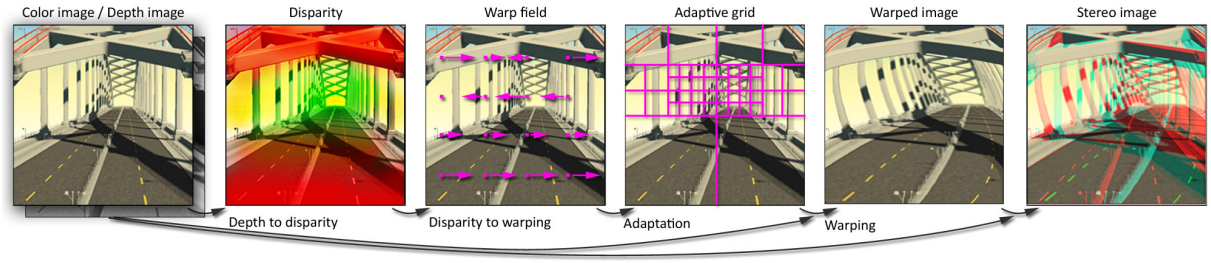


Figure 1: *Our image-based stereo view synthesis pipeline from left to right: We assume a rendered image with depth buffer, as well as a disparity map as the input of our method. If desired, the disparity map can be computed from the input depth image. Next, we build a warp field of this disparity mapping. This field is discretized adaptively: Areas with similar disparity are warped as large blocks, areas of different disparity are warped as small blocks. Finally, the input image and the new warped image are used as a stereo image pair, here, presented in anaglyph stereo.*

it's projection into the right eye's view. We call the mapping $f : \mathbb{R}^2 \to \mathbb{R}^2$ which maps every left image position $\mathbf{x}_{\text{left}}$ to its right image position $\mathbf{x}_{\text{right}}$ the *disparity mapping* from left to right. Further, we simply call the distance $\|\mathbf{x}_{\text{left}} - \mathbf{x}_{\text{right}}\| \in \mathbb{R}^+$ the *disparity* of $\mathbf{y}$.

The human visual system uses – besides several other mechanisms [Pal99, MIS04] – the disparity of image feature locations $\mathbf{x}_{\text{left}}$ and $\mathbf{x}_{\text{right}}$ to infer the depth of $\mathbf{y}$. Exploiting this principle, stereo displays achieve spatial vision by presenting two images with non-zero disparity which is then perceived as corresponding depths. There are various technologies to display individual images to each eye (shutter glasses, polarization filters, and others [MIS04]), but our work is independent of such display mechanisms.

Given a depth buffer, the simplest method to generate a disparity map is to apply a scale and bias to all values. In the case of a rendered scene, the depth can be directly output by the GPU, but our method does not rely on this particular feature and would support alternatively determined depth/disparity. While more complex approaches exist, scale and bias is easier to control and physical exactness is often less important than comfortable viewing.

We use a simple fragment program that applies a scale and bias to the depth in order to derive a disparity map. We adjusted our results in such a way, that both negative and positive parallax is present, as preferred by most viewers.

3.2. Pipeline

Our basic approach follows the pipeline depicted in Fig. 1. In order to facilitate the explanations, we will focus on how to produce a right image out of a given left image. Later in Section 3.5, we will extend this setting. We assume that the disparity mapping f is an input to this process and use it to convert a single image with depth information $I_{\text{left}}(\mathbf{x})$, into a pair of stereo images $I_{\text{left}}(\mathbf{x})$ and $I_{\text{right}}(\mathbf{x}) = I_{\text{left}}(f(\mathbf{x}))$.

Simply applying f in a pixel-wise fashion as done in previous approaches, can lead to holes and is not efficient to compute on a GPU, as it involves data scattering. Therefore, we represent f as a quad *grid*, i. e. a mapping from areas to areas instead of points to points [DER*10]. By doing so, we avoid holes and allow a parallel computation based on gathering instead of scattering, which is preferred for GPUs. We follow Didyk's [DER*10] approach: We start with a regular grid much coarser than the screen resolution and sample f at every vertex, we then warp this grid as textured quads into I_{right} and use I_{left} as a texture. While a grid-based approach avoids many holes, special considerations are required for the case of *occlusions* and *discoclusions*.

Occlusions occur when multiple locations $\mathbf{x}$ in I_{left} map to the same location in I_{right}. This happens for example, when a nearby object with a strong disparity covers a background object with low disparity in I_{right}. Indeed f might not have a unique inverse for some locations. However, such ambiguities can be resolved completely by using the depth information from $I_{\text{left}}(\mathbf{x})$: Whenever a pixel is written to $I_{\text{right}}(\mathbf{x})$, we compare its depth to the depth in $I_{\text{right}}(\mathbf{x})$ and omit the writing if its depth is bigger. In practice, this can be achieved using standard GPU depth buffering [DER*10].

Contrary to occlusions, disocclusions lead to holes because the originally hidden information is missing, but needed. Using the described grid warping, such holes are essentially filled with content from the input image [DER*10] by stretching the grid. A better solution, using multiple-image warping, is discussed in Section 3.4.

3.3. Adaptive Grid

While the previous approach succeeds in producing stereo image pairs (cf. the "Results" Section 4), it has two main drawbacks. First, if the image has many details in depth, a regular, coarse grid representation of f leads to undersampling and aliasing problems, i. e. low quality (cf. Fig. 7, Sec. 4). Second, just increasing the grid resolution (or keeping any fixed resolution), wastes an excessive amount of grid

vertices in areas which are essentially simple to warp using a low number of vertices, i. e. achieving low performance. We will now alleviate these two shortcomings by introducing an adaptive discretization of f.

As f is smooth over large areas, except at a few discontinuities, we construct a grid that *adapts* to the structure of f. We start from an initially regular grid (in practice, we start with a 32×32 grid to achieve enough parallelism, in theory one could start with 1×1 as well). The grid's quads are stored as a list of quad centers in an OpenGL vertex buffer object. A geometry shader traverses all these quads in parallel, and either outputs the same quad/center again, or refines this quad into four new quads/centers (cf. Fig. 2). This process is iterated until all quads are sufficiently refined and the structure well reflects the discontinuities in f.

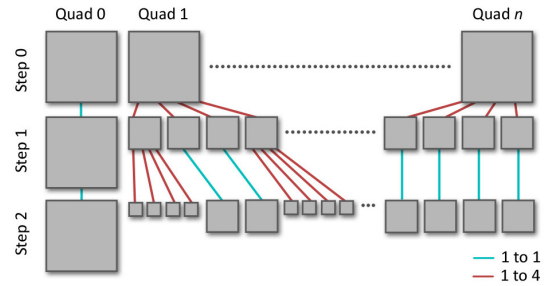


Figure 2: *Multiple quads (horizontal) subdivided in parallel using multiple steps (vertical). In every step, every thread produces either a single quad (1-to-1, blue) or four (1-to-4, red) new quads. In the next step, each quad is again processed in parallel. We repeat this until quads are pixel-sized.*

The decision whether a subdivision should be applied is based on the difference between minimal and maximal disparity inside the quad. If this difference is larger than some threshold four subquads are produced, else the quad is left unchanged. The output is captured in a second vertex buffer object using the OpenGL transform feedback extension. This subdivision process is iterated until the level 0 of 1×1-pixel-sized quads is reached in the regions where needed (hence, the number of steps depends logarithmically on the resolution of the input frame).

An alternative approach would be to directly refine a quad to many subquads, without recursion and without transform feedback. This leads to strongly varying output sizes (between one and several hundred vertices) which is not recommend for the geometry shader. Distributing the work amongst as-many-as-possible new threads after each subdivision is the preferred approach and allows for much more parallelism [MESD09].

Finally, when the subdivision is finished, we transform the vertex buffer object (VBO) quad centers back into a grid. For this, we use a second geometry shader that consumes quad centers and produces quads. f is evaluated for each corner of a quad and each quad is drawn to I_{right} using I_{left} as a texture, as described in the previous Section 3.2.

In order to avoid holes when disocclusions occur, it is important to realize that the grid vertices always fall on locations *between* two pixels (i. e. at level 0, a 1×1 quad maps to the corner of a pixel). We select the preferred pixel to fetch f and I_{left} based on its depth. That is, we fetch all four adjacent pixels around a vertex in I_{left} and use depth and disparity from the pixel with the smallest depth. By doing so, vertices adjacent to disocclusions effectively stretch across disocclusions and we can avoid holes across all levels.

3.3.1. Implementation Details

The position and level information for each quad is packed into an 8-bit RGB texture ($10 + 10$-bit position, 4-bit level).

To efficiently bound the amount of difference between minimal and maximal disparity inside a quad we use a min/max MIP-map. This map is similar to a common MIP-map, alas instead of storing the average, it stores the minimum and the maximum of all pixels below a pixel on higher levels. Such a map can efficiently be constructed in a parallel recursive fashion. Starting from level 0 at full resolution, a fragment program visits every pixel of the next-lower level and stores the minimum and the maximum of the four pixels from the lower level. This process is repeated until arriving at a single-pixel image, which, in our case, would store the minimum and maximum of all disparity values.

We set the subdivision threshold to 3 pixels which basically leaves only a low number of spurious single-pixel holes due to T-junctions, which occur if one quad is neighbor to a quad that is subdivided more. While a T-junction removal method could fix such problems, it usually generates again a higher and varying number of output vertices form the geometry shader. Doing so would significantly lower the geometry shader throughput, which is the bottleneck in our computation. We found the most efficient and simplest solution is, to just fill the undefined pixel via inpainting. In practice one can chose a random neighbor pixel in image space (cf. Fig. 3).

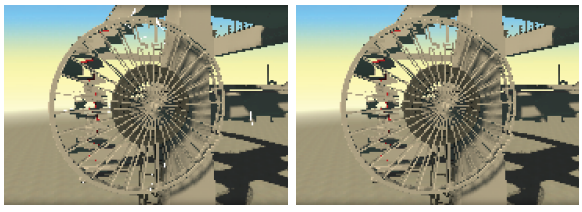


Figure 3: *We stop subdividing before reaching a pixel exact result (left) and fill the few remaining holes (right). Note, that this is an inset and pixel-sized holes are proportionally much smaller in multi-megapixel images.*

3.4. Using multiple images

Changing from a regular grid to an adaptive grid results in speed and quality improvements. Disocclusions remain the *only* visible artifact. By stretching the grid quads, the artifacts become less visible, but they can be perceived in certain configurations.

While disocclusions can ultimately not be solved without re-rendering, in this section, we will discuss how to use multiple images and multiple mappings to produce an improved stereo image pair.

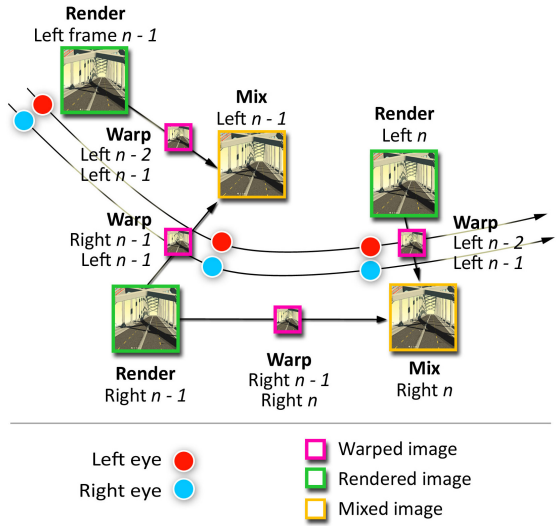


Figure 4: *Using multiple images to reduce disocclusions and improve quality. Consider the two eyes (red and cyan circle) of a moving observer in a virtual world (arrow). Ground truth would produce two images in each frame. Instead, we produce one frame only (green), warp (magenta) from the past and the other eye, and merge (yellow) according to the one with the lower error. To achieve convergence when slowing down or halting, we alternate the rendered and the synthesized image.*

We will use a previously rendered image I_{old} together with a mapping g which maps from the past view into the current view of the same eye (cf. Fig. 4). While f was defined to be a disparity mapping, g is not. Nonetheless, it is a mapping from $\mathbb{R}^2$ to $\mathbb{R}^2$ as well. g is also constructed rapidly via a fragment program which is executed on all depth-buffer pixels in parallel. These are unprojected from the old view and re-projected into the new view. The resulting 2D displacement is stored. As for f, g is not defined everywhere, for example if a location in the current frame was clipped in the previous frame.

We can now produce an alternative right stereo image $I_{\text{right}}(\mathbf{x}) = I_{\text{old}}(g(\mathbf{x}))$. I_{old} should be used whenever a disocclusion is present. To get the best result of both we carefully choose between the two sources. In practice, we use the stretching difference inside a quad: If a quad undergoes varying stretching, it is likely to cause a disocclusion (it "tears up" the space) and should therefore not be used. Precisely, we use a *preference* operator w, arriving at

$$I_{\text{right}}(\mathbf{x}) = \frac{w(f)(\mathbf{x}) \cdot I_{\text{left}}(f(\mathbf{x})) + w(g)(\mathbf{x}) \cdot I_{\text{old}}(g(\mathbf{x}))}{w(f)(\mathbf{x}) + g(f)(\mathbf{x})},$$

with

$$w(h)(\mathbf{x}) : (\mathbb{R}^2 \to \mathbb{R}^2) \to (\mathbb{R}^2 \to \mathbb{R}).$$

The operator w turns the (disparity) mapping h into a spatially varying preference for that mapping.

Although there is no guarantee, that all occlusions will be resolved. This strategy performs rather well because a disocclusion in one mapping will often not be a disocclusion in another. Following the same strategy, we can also avoid the T-junction holes nearly completely. Only such holes that are present in *both* images remain holes, which is never the case in practice when relying on a three pixel threshold in a multi-megapixel image.

3.5. Convergence

One final step can further improve the result: Instead of always rendering the left eye view and creating a right eye view, we can *swap* the eye roles and either warp from left to right or right to left. Swapping eyes in every frame does not lead to a strong improvement as long as the viewer is moving, nonetheless, also no temporal artifacts are introduced. However, already in this setting, if the speed of the motion decreases, w will prefer the past image, and ultimately, when no animation is present, w will always pick the past right eye for the current right eye and the past left eye for the current left eye, i. e. the result converges to the static reference.

In order to further improve the quality in the case the camera is moving, instead of toggling, it is best to choose the most distant eye view from the previously rendered. In such a way we minimize the potential disocclusion. In order to visualize the advantage of this choice, one can imagine a constant panning movement. If the left eye always falls on the old position of the right eye, a toggling would be harmful, as it would lead to the same view being rendered twice. Choosing the most distance view eases the handling of disocclusion. In this particular case, in combination with the operator w, our algorithm even produces the reference result, although the camera is no longer static.

4. Results

In this section we evaluate quality and performance of our approach. We used an NVIDIA Quadro FX 5800.

To test our approach, we have chosen mostly architectural models because they represent an excellent stress test with

many occlusions, disocclusions and fine details. All models are rendered using shadow mapping, per-pixel deferred shading, fog, depth of field and screen-space ambient occlusion. With such a set-up it takes around 40 ms to produce a frame. We excluded the computation of the disparity from all timings as we assume it to be an input of our method.

We compare our method to three other approaches. First, straightforward mapping of a 1×1 grid, including handling of occlusions in the same way as this paper does. This approach is our *reference* solution in terms of speed. Using our method by morphing only one image we can only approach the quality of such solution. An improvement is possible using more views as described in Sec. 3.4. Second, we show that our method produces better results in terms of speed and quality than using pixel-wise re-projection. We also compare our method to Didyk et al.'s [DER*10] approach for temporal upsampling. Their method addressed viewpoint synthesis in time, using a carefully optimized GPU implementation as well. It is significantly faster than the reference approach, but has lower quality. We will substantially improve upon this method in terms of quality, and in some cases even in terms of speed.

Quality and Performance To show the importance of using an adaptive approach we compared our one view morphing method to the naïve, reference solution. Although we cannot improve the quality, we can bound an error by setting the subdivision threshold properly. Doing so, the solutions of both methods become indistinguishable but due to the adaptivity, our solution is several times faster.

In Fig. 7 we compare the performance and the quality of our approaches as well as Didyk et al.'s method to ground truth rendering. First, we see how our approach speeds up the process of producing stereo content compared to rendering two frames. On average, for all scenes used for our experiments, the morphing of one frame in resolution 2048×1024 takes around 7 ms.

Second, our method achieves quality similar to the ground truth, while Didyk et al.'s approach falls short in doing so for complex details (spikes, ghosting). In particular, when comparing to the trivial approach (Fig. 8) of mapping individual pixels and filling the holes using pull-push, the quality is worse and the performance is three times lower. This is easy to see, as warping a grid of vertices which form a small subset of all pixels in the image is obviously faster than warping all pixels. This performance difference underlines the importance of supporting modern fine grained parallelism (i. e. gathering) over straightforward approaches which require scattering.

Third, we see how the use of multiple images avoids disocclusions and improves the quality by comparing the two rightmost columns. This is most visible for the "Antenna" scene in the second row, where the thin features are stretched across disocclusions when using only a single image. As our approach is orthogonal to the used surface representation, we can apply our technique also directly to iso-surface raycasting [Lev88] (last row).

Adaptation Quality Further, we seek to illustrate the influence of the subdivision threshold by keeping all parameters fixed and varying only this threshold. In Fig. 5, we show

Figure 5: *Decreasing (resp. increasing) the threshold generates a higher (resp. lower) grid resolution, therefore lower (resp. higher) speed but also higher (resp. lower) quality.*

high, medium and low-quality thresholds, the respective subdivision, as well as some details that represent typical problems also encountered with a trivial approach (Fig. 8).

Analysis In Fig. 6, the variation of performance over time for the reference, Didyk et al.'s and our method is plotted for the "Crane" scene. We see, how our method has varying efficiency over time. This is because the adaptation creates a varying number of quads in our grid. However, it is almost never slower than previous work, at much higher quality, as discussed in the previous paragraph. Tighter bounding of this time interval is desirable in interactive applications such as games and remains future work.

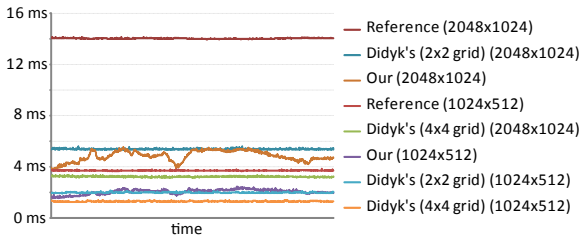


Figure 6: *Variation of performance over time for several different strategies. Although our performance varies due to the adaptivity, it is nearly as high as for Didyk's approach but at a quality comparable to the reference solution.*

5. Discussion and Limitations

Similar to many other upsampling methods, our approach is limited to non-transparent surfaces. We do not account for view dependent-effects such as specular highlights.

The improvement when using previous frames (Sec. 3.4) depends on the camera path. In case of camera movement

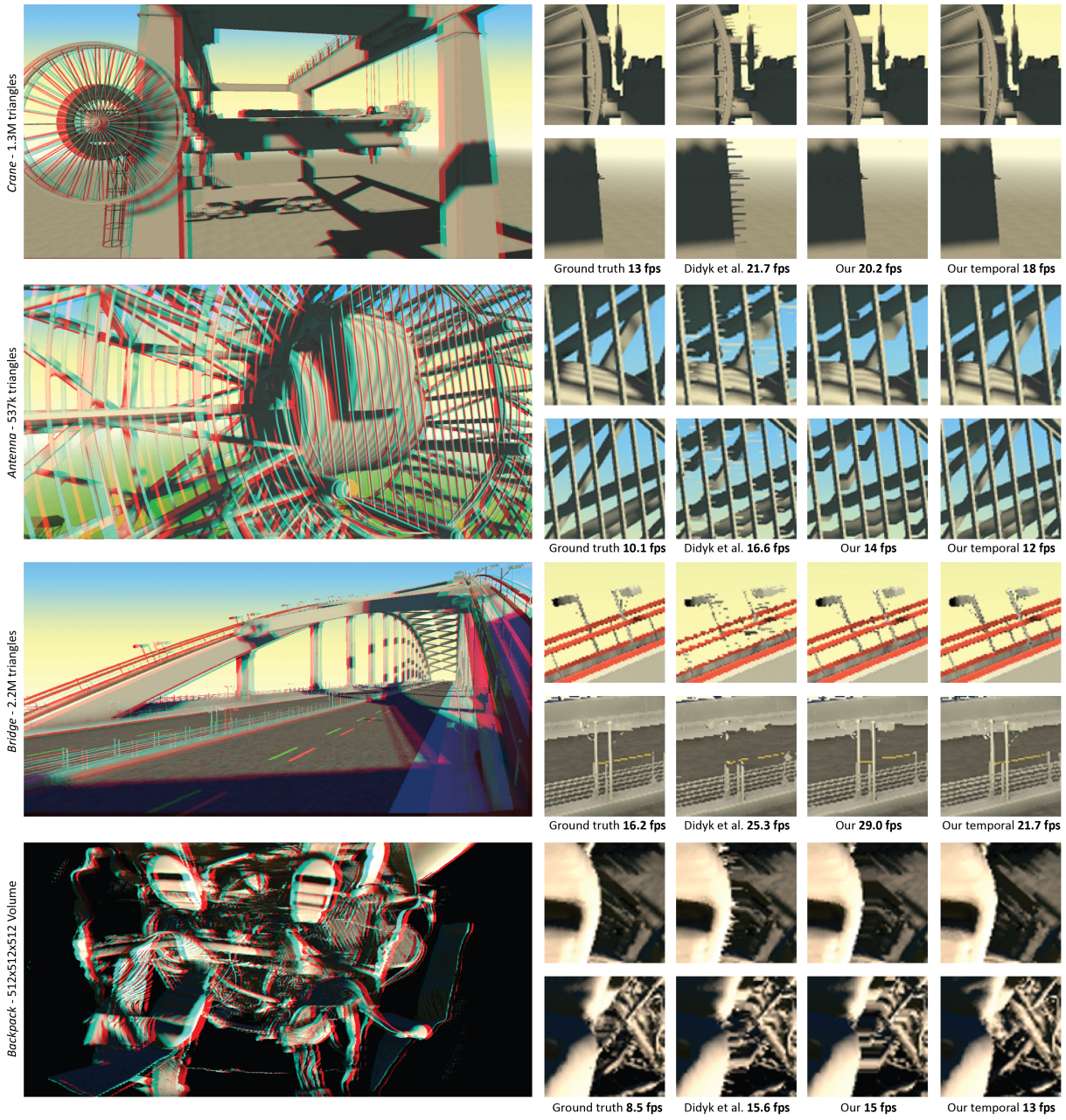


Figure 7: *Results produced by our algorithm (Left) for different scenes in resolution 2048×1024, presented in anaglyph stereo. On the right, we show scene details computed using four different approaches: Ground truth; Didyk et al.'s method; Ours using only single images; Ours using multiple images. We achieve similar quality to ground truth at a performance similar to Didyk et al.'s method (see the fps insets).*

in the plane to which the eye axis is normal, no additional information is won, but such movements are less likely than e. g. human walking animations. Put in another way, human eyes are placed horizontal to each other and not vertically because of the movements performed by humans [Ros74]. In future work, more advanced view selection techniques are worth investigating.

Lacking a suitable output device, we were not able to test our method for generating more than two views out of

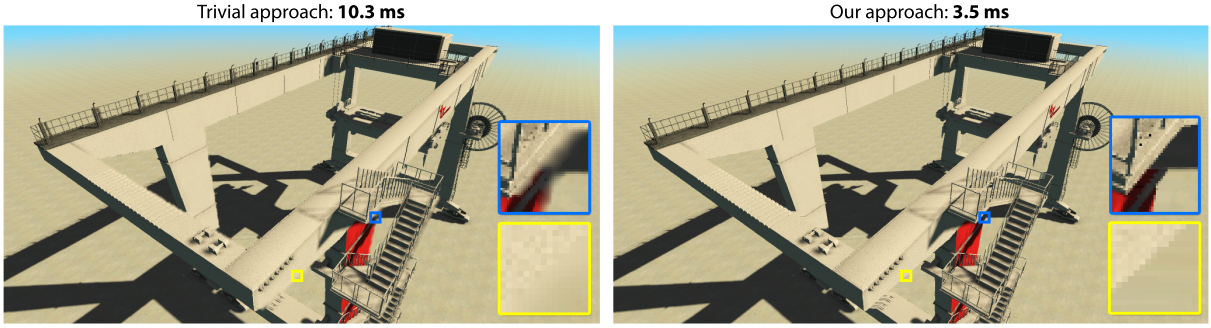


Figure 8: *Using pixel-wise re-projection (trivial approach, below-reference) results in many holes, that have to be filled using pull-push which leads to blur. At the same time, the performance is approximately three times lower than for our approach.*

one. However, the time-benefit of image-based upsampling would be even more pronounced. Also, we envision upsampling in time as well as in stereo and other image-based reuse e. g. for anti-aliasing or motion blur.

6. Conclusion and Future Work

This work described an approach to upsample a stream of monocular images with depth information to a pair of stereo images streams, exploiting modern GPUs and human perception. We demonstrate its application to a number of problems, in which the approach drastically reduces the rendering time compared to rendering an image pair. The approach is independent of the underlying surface representation and can be easily integrated into existing software as a post-process to deliver high-quality stereo-image pairs.

References

[CW93] CHEN S. E., WILLIAMS L.: View interpolation for image synthesis. In *Proc. SIGGRAPH* (1993), pp. 279–288.

[DEF*07] DOMONKOS B., EGRI A., FÓRIS T., SZIRMAY-KALOS L., TAMÁS J.: Isosurface ray-casting for autostereoscopic displays. In *WSCG, Short Papers* (2007).

[DER*10] DIDYK P., EISEMANN E., RITSCHEL T., MYSZKOWSKI K., SEIDEL H.-P.: Perceptually-motivated real-time temporal upsampling of 3D content for high-refresh-rate displays. *Comput. Graph. Forum (Proc. Eurographics) 29*, 2 (2010), 713–722.

[HZ00] HARTLEY R. I., ZISSERMAN A.: *Multiple View Geometry in Computer Vision*. Cambridge University Press, 2000.

[KKS08] KNORR S., KUNTER M., SIKORA T.: Stereoscopic 3D from 2D video with super-resolution capability. *Signal Processing: Image Communication Vol. 23*, 9 (Oct. 2008), 665–676.

[Lev88] LEVOY M.: Display of surfaces from volume data. *IEEE Comput. Graph. Appl. 8*, 3 (1988), 29–37.

[LGJA09] LIU F., GLEICHER M., JIN H., AGARWALA A.: Content-preserving warps for 3D video stabilization. *ACM Trans. Graph (Proc. SIGGRAPH) 28* (2009), 44:1–44:9.

[MESD09] MEYER Q., EISENACHER C., STAMMINGER M., DACHSBACHER C.: Data-parallel hierarchical link creation for radiosity. In *Proc. EPGV* (2009), pp. 65–69.

[MHM*09] MAHAJAN D., HUANG F.-C., MATUSIK W., RAMAMOORTHI R., BELHUMEUR P.: Moving gradients: A path-based method for plausible image interpolation. *ACM Trans. Graph. (Proc. SIGGRAPH) 28*, 3 (2009), 42:1–42:11.

[MIS04] MEESTERS L., IJSSELSTEIJN W., SEUNTIENS P.: A survey of perceptual evaluations and requirements of three-dimensional TV. *IEEE Trans. Circuits and Systems for Video Technology 14*, 3 (March 2004), 381–391.

[MMB97] MARK W. R., MCMILLAN L., BISHOP G.: Post-rendering 3D warping. In *Proc. ACM I3D* (1997), pp. 7–16.

[Mor76] MORLAND D. V.: Computer-generated stereograms: a new dimension for the graphic arts. *SIGGRAPH Comput. Graph. 10*, 2 (1976), 19–24.

[NSL*07] NEHAB D. F., SANDER P. V., LAWRENCE J., TATARCHUK N., ISIDORO J.: Accelerating real-time shading with reverse reprojection caching. In *Proc. Graphics Hardware* (2007), pp. 25–35.

[Pal99] PALMER S. E.: *Vision science : Photons to phenomenology*. MIT Press, Cambridge, Mass., 1999.

[Ros74] ROSS J.: Stereopsis by binocular delay. *Nature 248* (March 1974), 363–364.

[SGH*01] SAWHNEY H. S., GUO Y., HANNA K., KUMAR R., ADKINS S., ZHOU S.: Hybrid stereo camera: An IBR approach for synthesis of very high resolution stereoscopic image sequences. In *Proc. SIGGRAPH* (2001), pp. 451–460.

[SLW*08] STICH T., LINZ C., WALLRAVEN C., CUNNINGHAM D., MAGNOR M.: Perception-motivated interpolation of image sequences. In *Proc. APGV* (2008), pp. 97–106.

[WDP99] WALTER B., DRETTAKIS G., PARKER S.: Interactive rendering using render cache. In *Proc. EGSR* (1999), pp. 19–30.

[Whe38] On some remarkable, and hitherto unobserved, phenomena of binocular vision. *Roy. Soc. London Phil. Trans.* (1838).

[Wol98] WOLBERG G.: Image morphing: A survey. *The Visual Computer 14*, 8 (1998), 360–372.

[ZHQ*07] ZHANG G., HUA W., QIN X., WONG T.-T., BAO H.: Stereoscopic video synthesis from a monocular video. *IEEE Trans. Visualization and Comput. Graph. 13*, 4 (2007), 686–696.

[ZKU*04] ZITNICK C. L., KANG S. B., UYTTENDAELE M., WINDER S., SZELISKI R.: High-quality video view interpolation using a layered representation. In *Proc. SIGGRAPH* (2004), pp. 600–608.