EUROGRAPHICS 2011 / M. Chen and O. Deussen
(Guest Editors)

Volume 30 (2011), Number 2

Supplemental Material: Scalable Remote Rendering with Depth and Motion-flow Augmented Streaming

Dawid Pajak[1,2] Robert Herzog[2] Elmar Eisemann[3] Karol Myszkowski[2] Hans-Peter Seidel[2]

[1]West Pomeranian University of Technology, Szczecin, Poland
[2]MPI Informatik, Saarbrücken, Germany
[3]Telecom ParisTech, Paris, France

Abstract

The purpose of this document is to provide complementary performance and quality results described in Section 5 of the paper. Most important aggregated figures are presented in the main paper while here we focus on each component and performance setting separately. At the end we show the 3D stereo vision and spatio-temporal upsampling quality comparison between our scheme and H.264.

Scene	Resolution	Server											Client					
		Brute-force			Our Method													
		T_{shade}	T_{h264}	Fps	Upsample	T_{geom}	T^{low}_{shade}	T_{prep}	T_{edge}	T_{enc}	T^{low}_{h264}	Fps	Speedup	T^{low}_{h264-1}	$T_{predict}$	T_{dec}	T_{dif}	T_{up}	Fps
SIBENIK	SVGA	57.2	32.3	10.7	2×2	4.1	16.0	2.6	1.8	10.3	12.5	21.1	× 2.0	2.0	0.9	9.9	1.7	1.2	63.7
					4×4	4.1	4.5	2.2	1.6	17.5	9.1	25.6	× 2.4	1.0	0.9	17.1	1.3	1.2	46.5
	HD 720p	106	46.0	6.3	2×2	5.9	29.0	3.7	3.6	16.4	18.3	13.0	× 2.1	3.1	1.7	16.1	2.2	2.3	39.4
					4×4	6.0	8.5	3.2	3.0	17.1	9.7	21.1	× 3.3	2.0	1.7	16.6	1.5	2.3	41.5
	HD 1080p	251	73.7	3.0	2×2	10	67.5	7.3	7.9	31.0	32.8	6.4	× 2.1	7.2	3.9	30.5	3.4	4.8	20.1
					4×4	10.7	18.4	7.3	6.6	33.5	13.6	11.1	× 3.7	2.2	3.7	32.5	3.4	4.8	21.5
SPONZA	SVGA	63.0	32.4	10.0	2×2	4.6	16.7	2.6	1.9	20.8	12.7	16.9	× 1.7	2.0	1.0	20.8	1.5	1.2	37.7
					4×4	4.6	4.5	2.3	1.6	20.5	9.2	23.4	× 2.3	1.0	1.1	21.3	1.5	1.2	38.3
	HD 720p	120	45.0	5.8	2×2	6.6	31.9	4.0	3.6	21.5	19.2	11.5	× 2.0	3.0	2.0	21.1	2.1	2.2	32.9
					4×4	6.6	8.5	3.9	3.0	25.5	9.7	17.5	× 3.0	2.1	1.7	25.1	2.1	2.2	30.1
	HD 1080p	271	76.4	2.8	2×2	11.0	71.4	8.0	7.9	40.9	33.2	5.8	× 2.1	7.1	4.1	39.7	3.9	4.8	16.8
					4×4	11.9	18.8	7.8	6.6	43.0	14.0	9.8	× 3.5	2.1	3.8	42.1	3.9	4.7	17.7
FAIRY	SVGA	31.6	31.0	14.9	2×2	4.4	8.7	2.4	1.9	19.0	12.1	20.6	× 1.4	1.9	0.9	17.5	1.4	1.1	43.9
					4×4	4.4	2.8	2.2	1.6	15.0	9.2	28.4	× 1.9	1.0	0.9	14.1	1.3	1.1	54.3
	HD 720p	51.2	44.4	9.9	2×2	5.8	15.2	4.3	3.6	17.3	18.4	15.5	× 1.6	3.0	1.7	16.7	2.6	2.1	38.3
					4×4	5.9	4.7	4.3	3.0	22.7	9.6	19.9	× 2.0	2.1	1.7	20.8	2.6	2.1	34.1
	HD 1080p	107	72.2	5.3	2×2	9.3	28.7	7.5	7.9	34.7	30.3	8.4	× 1.6	7.5	3.9	32.1	3.6	4.7	19.3
					4×4	9.4	7.3	8.7	7.0	41.0	13.6	11.5	× 2.2	2.0	5.9	38.8	2.8	4.7	18.5

Table 1: *Our Server and Client timings (in msec) for different target resolutions:* 800×600 *(SVGA),* 1280×720 *(HD 720p),* 1920×1080 *(HD 1080p) compared to reference (full-res.) rendering* T_{shade} *and following video-encoding step* (T_{h264}) *using H.264 with disabled frame delay. The geometry-processing costs* (T_{geom}) *are the same for reference and our method. However, in our examples rendering costs are dominated by frame size due to complex pixel shading and hence, we give timings for different upsampling factors (2 and 4). In our system the server-side timings comprise: the geometry pass* (T_{geom}) *including the rendering preprocess (updating animation, shadow maps, etc.), the (deferred) low-res. pixel shading* (T^{low}_{shade}), *the preparation pass simulating the client steps* (T_{prep}), *which is initiated before the edge detection and encoding (i.e., previous depth/motion frame warping and edge diffusion), the edge selection* (T_{edge}) *which involves computing the perceptual edge significance in the low-res. shading image (up to this point all steps are performed on the GPU), our edge encoding including entropy coding on the CPU* (T_{enc}), *the H.264 low-res. frame encoding* (T^{low}_{h264}). *On the client side we perform: low-res. H.264 stream decoding* (T^{low}_{h264}), *previous depth/motion frame warping and edge prediction* ($T_{predict}$) *on GPU, edge image decoding* (T_{dec}), *edge diffusion* (T_{diff}) *followed by upsampling the low-res. video-frame* (T_{up}). *Note that several steps in the system pipeline are independent and could be executed in parallel.*

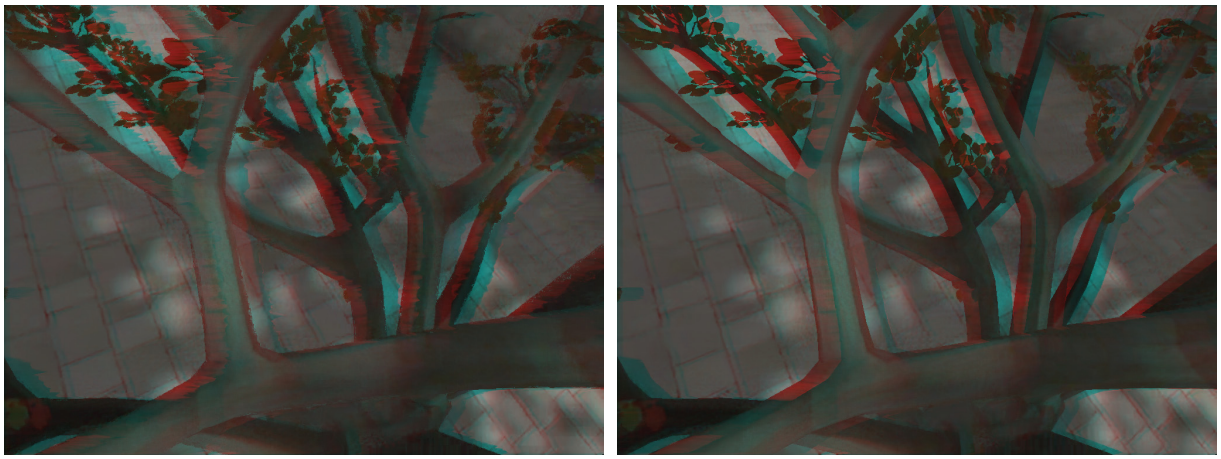

Figure 1: *3D stereo warping quality comparison. Precise depth values along geometric discontinuities are crucial for correct and high-quality image warping. Left image shows warping result using frame and depth encoded with H.264 codec at 4 Mbit/sec. Imprecise edge representation generates visible distortions in reprojected image. Our solution (right image) addresses this issue by employing a custom depth/motion compression scheme that prioritizes values along the edges (3.8 Mbit/sec).*

Figure 2: *Spatio-temporal upsampling quality comparison. Here we show another type of visual distortion when using a DCT-based video codec for depth/motion compression. Decompressed motion vectors include small errors that cause temporal reprojection to warp pixels to bad locations. This effect propagates over time and results in noisy pixels and smearing in high contrast areas. As we store precise values at depth discontinuities our approach (right image) is not affected by the aforementioned artifact. Bandwidth settings are the same as in Fig. 1.*

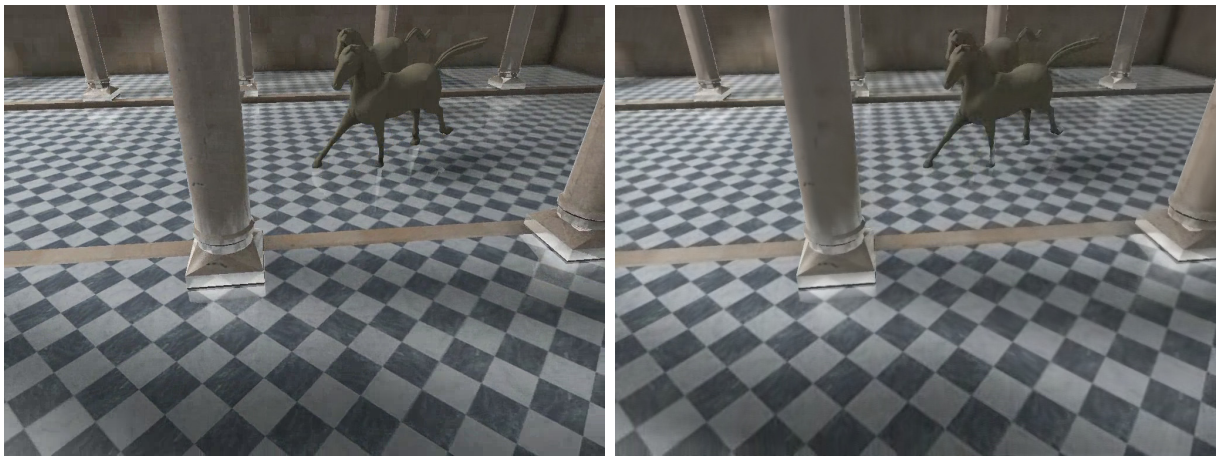

Figure 3: *Final video quality comparison. Here we show a scene featuring most difficult configuration for our architecture: dynamic lighting, dynamic geometry and mirror reflections on the floor and pillars. Our method (right image) still delivers acceptable quality with 2.8 Mbit/sec when comparing it to the reference encoded with H.264 (left image) with 3.0 Mbit/sec. Please note that our upsampled frame in particular the view-dependent reflections in the pillar appear more blurry. However, this is less noticeable when watching the accompanying video sequences.*