MADmax: A 1080p Stereo-to-Multiview Rendering ASIC in 65 nm CMOS based on Image Domain Warping

Michael Schaffner*†, Pierre Greisen*†, Simon Heinzle†, Frank K. Gürkaynak*, Hubert Kaeslin*, Aljoscha Smolic†
*ETH Zürich, 8092 Zürich, Switzerland
†Disney Research Zurich, Switzerland

***Abstract*— In this paper, a video rendering ASIC for *multiview automultiscopic displays* using an *image domain warping* approach is presented. The video rendering core is able to synthesize up to nine interleaved views from full-HD (1080p) *stereoscopic 3D* input footage. The design employs *elliptical weighted average* (EWA) splatting to perform the image resampling. We use the mathematical properties of the Gaussian filters of EWA splatting to analytically integrate display anti-aliasing into the resampling step. The use of realistic assumptions on the image transformation enable a hardware architecture that operates on a video stream in scan-line fashion and that does not require an off-chip memory. The ASIC, fabricated in a 65 nm CMOS technology, runs at 260 MHz and is able to deliver 28.7 interleaved full-HD (1080p) frames per second with eight views enabled. It has a core power dissipation of 550 mW and its complexity is 6.8 MGE, including 4.36 MBit SRAM macros.**

I. Introduction

Multiview autostereoscopic displays (MADs [1]) that provide a glasses-free 3D experience have recently become popular. Unfortunately, such displays need images from multiple view points, and automatic *multiview* (MV) content creation methods have therefore been researched extensively. *MV synthesis* is an important technique that addresses the problems of content creation and transmission for such displays. The idea is to generate all required views from a lower number - typically two for *stereo 3D* (S3D) - at the display. Common MV synthesis methods are based on *depth image based rendering* (DIBR) such as in [2]–[5], where a dense depth map of the scene is used to reproject the image to new viewpoints. Although physically correct, this approach requires accurate depth maps and additional inpainting steps. An alternative S3D to MV conversion concept is suggested by [6], and is based on *image domain warping* (IDW) [7]. This technique is promising as it does not rely on pixel dense depth, but only on sparse image features. Further, no inpainting is needed which is still an algorithmic/computational limitation of DIBR.

The input to the IDW processing pipeline is the S3D footage (left and right images) which is analyzed in order to reveal sparse image features (such as point correspondences, vertical lines and saliency information) in a first step. Those features are then used to calculate two warps - one for each input image. These warps describe the (nonlinear) transformation of the input images to a viewing position centered between the two original views. The new views are then generated by first inter- and extrapolating the two warps to the desired view positions; and secondly by resampling the S3D input according to those interpolated warps. Finally, the generated views are interleaved in such a way that they can be displayed on the MAD.

The ASIC presented here implements the warp interpolation, image resampling and interleaving steps (Figure 1) of this IDW approach. The circuit can either be used as part of a larger system with online image analysis and warp generation, or in a stand-alone configuration where the preprocessing is performed on the encoder side and the warps are transmitted together with the content (S3D+warp) [8]. This is similar to DIBR configurations where a depth map is transmitted together with the content (S3D+depth).

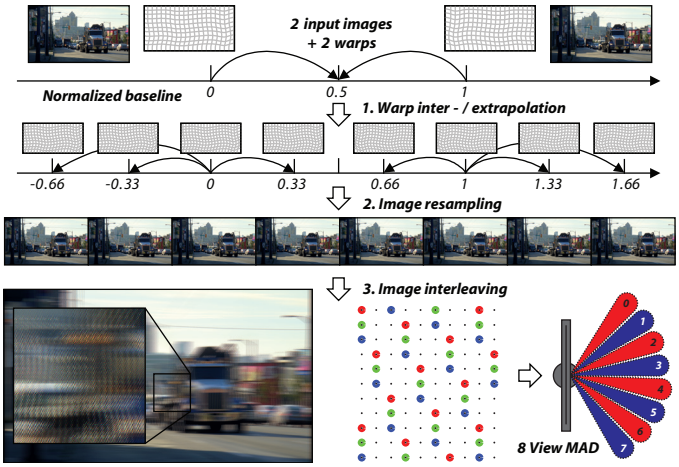

Fig. 1. IDW multiview rendering concept with 8 views: First, the warps for the desired view positions are calculated via linear interpolation of the input warps. The input images are resampled according to those warps in a second step. Finally, they are interleaved such that they can be displayed on an MAD. The interleaving pattern of one view is shown at the bottom, where the color indicates the associated subpixel. Note that in lenticular-based MADs individual subpixels are mapped to different views.

Summary of Contributions: We provide a design and an implementation of a hardware architecture for real-time MV rendering. The architecture is based on our previous image warping implementations [9], [10], which use *elliptical weighted average* (EWA) splatting to resample the images. In contrast to [9], [10], the whole rendering pipeline has been redesigned to meet the requirements of MV rendering. In particular, display anti-aliasing and pattern based filter-evaluation capabilities have been added, the throughput has been increased, and the framebuffer caching architecture was adapted to the sparse nature of typical sampling patterns. Furthermore, additional circuitry for warp pre-processing has been included. The resulting ASIC was fabricated in 65 nm CMOS technology (UMC) and achieves a throughput of 28.7 fps (1080p, with 8 views) with a core power dissipation of 550 mW.

II. MV Synthesis using Image Domain Warping

The employed MV rendering concept of [6] is shown in Figure 1, and can be roughly divided into three steps. The two input warps map the two input images to a new view position on the baseline which is in between the two original views. In a first step, those two warps are then used to linearly inter- and extrapolate as many new warps as required (in this case an example with 8 views is shown). The warps are then applied to the corresponding input image in a second step, and finally the resulting views are interleaved in one output image in a third step. The algorithms employed in the those three steps are summarised in the following subsections.

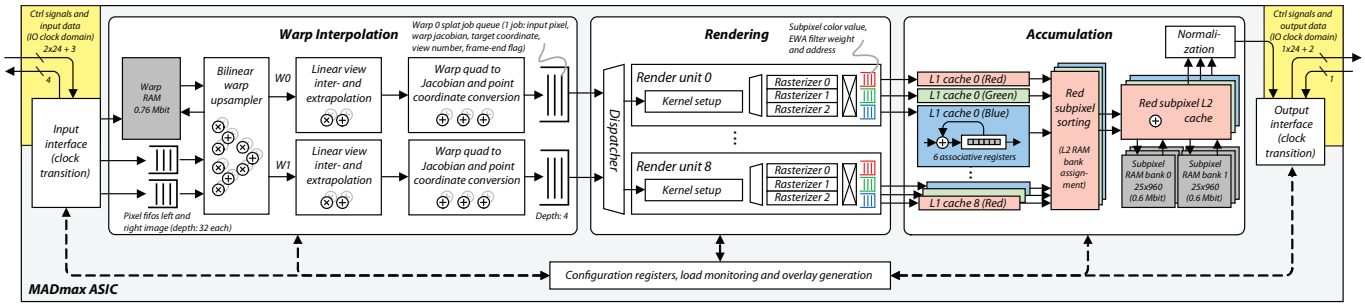

Fig. 2. The MADMAX ASIC consists of three main stages. The *Warp Interpolation* stage preprocesses the incoming warp data, groups the calculated coordinates with the input image data and dispatches those packets or *splat jobs* to the *Rendering* stage. The splats are assigned to the correct render unit (each unit is responsible for one view), where the kernels are set up and rasterized on a programmable sampling pattern. The *Accumulation* receives the subpixel values and accumulates them in a frame buffer. Note that all I/Os of the chip work in a clock domain which is $4\times$ slower than the core clock.

A. Warp Interpolation

An image warp can be described using a non-linear, two dimensional mapping $m(\mathbf{u}) : \mathbf{u} \in \mathbb{R}^2 \rightarrow m(\mathbf{u}) \in \mathbb{R}^2$. $\mathbf{u}$ is the two-dimensional coordinate in the input image, and a linearized index k is used to indicate when this coordinate is a discrete sampling point $\mathbf{u}_k$ (i.e. a pixel position). Let $m^{\alpha_0}(.)$ denote the input warp which maps the corresponding input image to the relative position α_0 on the normalized baseline. The warp is linearly inter- and extrapolated using

$$m^{\alpha_n}(\mathbf{u})_k = \left(1 - \frac{\alpha_n}{\alpha_0}\right) \cdot \mathbf{u_k} + \frac{\alpha_n}{\alpha_0} \cdot m^{\alpha_0}(\mathbf{u}_k), \tag{1}$$

where α_n is the desired position on the baseline of view n. For most applications, the image warps have a lower resolution than the images. In this project, warps with a resolution of 180×100 are used. Therefore, the input warps are first up-sampled using bilinear interpolation prior to the actual view interpolation.

B. Image Resampling and Anti-Aliasing

The images are resampled using EWA splatting, which is a *forward-mapping* method. This has the advantage that no warp inversion is required as the warps are generated in forward format in this application [6]. The EWA framework uses Gaussian filter kernels and the Jacobian of the image warp as a local deformation measure in order to calculate the footprint of an input image pixel in the output image. The input pixels thus correspond to Gaussian *splats* in the output image, which are rasterized within a bounding box and accumulated in a frame buffer. Since Gaussians are closed among themselves and under affine transformations, an anti-aliasing filter for the output image sampling grid can be easily incorporated analytically. A short summary is given below (for a complete derivation we refer to [9], [10]).

The EWA Filter Kernel: Let w_k be the input pixel value at position $\mathbf{u}_k \in \mathbb{N}^2$, where k is a linear pixel index. Without loss of generality we assume w_k to be scalar here. As before, $m(\mathbf{u})$ denotes the image warp. Let J_k be the Jacobian of the warp at pixel position $\mathbf{u}_k$. The EWA kernel is characterized by the covariance matrix

$$\Sigma_k = J_k V_i J_k^T + V_{aa} = C_k + V_{aa} \tag{2}$$

in the target image domain, where the first term is the transformed interpolation kernel, and the second term is the anti-aliasing kernel. $V_i = \mathrm{diag}(\sigma_i^2, \sigma_i^2)$ and $V_{aa} = \mathrm{diag}(\sigma_{aa}^2, \sigma_{aa}^2)$ are diagonal covariance matrices that parameterize the interpolation and anti-aliasing filters. The weight of the Gaussian filter at the discrete position $\mathbf{x}_j \in \mathbb{N}^2$ in the output image is calculated as

$$\rho_{jk} = \frac{|J_k|}{2\pi\sqrt{|\Sigma_k|}} e^{\left(-0.5\left(\mathbf{x}_j - m(\mathbf{u}_k)\right)\Sigma_k^{-1}\left(\mathbf{x}_j - m(\mathbf{u}_k)\right)\right)}, \tag{3}$$

and is multiplied with the pixel value w_k.

Post-Normalization: The individual transformation of the input pixels can lead to normalization problems in the output image, and thus a so called post-normalization is performed. To do so, the filter weights ρ_{jk} are accumulated along with the pixel values $\rho_{jk} \cdot w_k$. At the end of the rendering process the output image pixels p_j are calculated by dividing the accumulated values by the corresponding weight $p_j = (\sum_{\forall k} \rho_{jk} \cdot w_k)/(\sum_{\forall k} \rho_{jk})$.

Parametrization: In [9] it is shown that for a regular, quadratic sampling grid the filter parametrization $\sigma_i \approx 0.39$ leads to the optimal L2 fit of a Gaussian to the ideal low-pass filter in the frequency domain. In the same paper it is also shown how σ_{aa}^2 can be chosen in an adaptive way in order only perform anti-aliasing when needed. For the application at hand, the covariance matrices are diagonally dominant, and therefore we can use the *simplified adaptive* scheme which boils down to the following threshold rule:

$$\Sigma_k = \begin{bmatrix} \max(C_k^{00}, V_{aa}^{00}) & C_k^{01} \\ C_k^{10} & \max(C_k^{11}, V_{aa}^{11}) \end{bmatrix}, \tag{4}$$

where the superscripts are the element indices and $\sigma_i = \sigma_{aa} = 0.39$.

Display Anti-Aliasing: The resampled views are interleaved into one output image according to a special interleaving pattern (like in Figure 1), such that they can be displayed simultaneously on a MAD. Proper care must be taken in order to prevent aliasing, as shown in [11]. The filters are generally non-separable, and a high order is required to approximate the intricate shape of the passband. But [11] also noted that for natural images, the benefit of such filters is rather small. As a result, simpler separable filters that lead to visually pleasing results could be used as well. In this work we use the closedness of Gaussians in order to incorporate a Gaussian display pre-filter analytically into the EWA filter kernel. Instead of using adaptive EWA splatting with $\sigma_{aa} = 0.39$ we adapt this value with the *density* $d = 1/|\Lambda|$ of the *display sampling lattice* Λ

$$\sigma_{disp}^2 = \sigma_{aa}^2/d = \sigma_{aa}^2 \cdot |\Lambda|, \tag{5}$$

where σ_{disp}^2 is now used in place[1] of σ_{aa}^2. The display we used for experiments has a density of $1/8$, which results in $\sigma_{disp}^2 \approx 1.22$.

[1]Note that anisotropic anti-aliasing is also possible by defining two different variances on the diagonal $V_{aa} = \mathrm{diag}(\sigma_{disp1}^2, \sigma_{disp2}^2)$. This can be useful in applications where EWA splatting is used to directly render other interleaved images such as column or row interleaved stereo images.

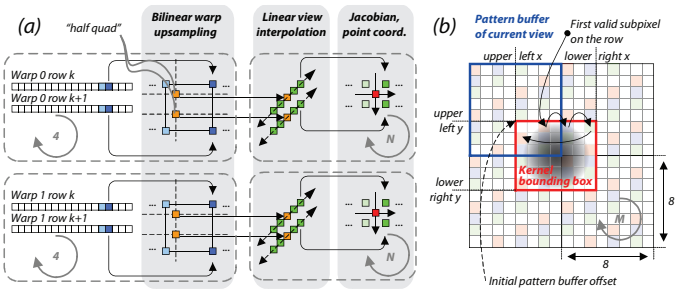

Fig. 3. a) The warp interpolation stage first rescales the 180×100 warps using bilinear interpolation. Second, the warps are interpolated to the desired views and third, the Jacobian and the target coordinates are calculated. b) Rasterization process: the rasterizers of a render unit only evaluate the filter kernels at the effectively required positions.

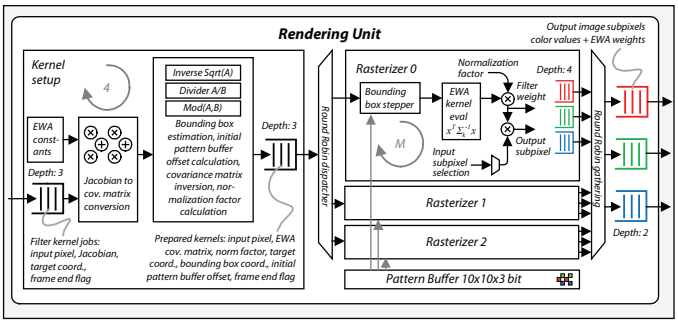

Fig. 4. One render unit consists of a kernel setup unit which prepares the filter kernels for efficient evaluation in one of the three rasterizer units. The throughput of the setup unit and the rasterizers is matched when the average number of rasterized subpixels per splat is $M \approx 12$ (Figure 5a).

III. Architecture

Figure 2 shows the top level VLSI architecture of our ASIC. It consists of three main stages. The two input images and warps are streamed in through the input interface in an interleaved manner. The coordinates for the synthetic views are generated by the *warp interpolation* stage on the fly, and are paired with the correct view numbers and input pixels. Those packets are then passed on to the *rendering* stage where they are dispatched to the correct render unit. There are nine render units and each is allocated a specific view number $\in [0, ..., 8]$). The rendered subpixels are sent to the *accumulation* stage, where a two-level caching scheme is employed to combine them and form the complete output image.

A. Input and Output Interfaces

For the following throughput calculations we use a core clock frequency of 300 MHz. The I/Os of the chip operate at a $4\times$ slower clock frequency (75 MHz) than the core. The chip has three 24 bit RGB ports – two at the input and one at the output. Each port provides a bandwidth of 1.8 Gbit/s, which is enough to transfer 1080p video at 30 fps (1.49 Gbit/s). The warp data rate is relatively modest and amounts to only $2 \times 180 \times 100 \times 21$ bit $\times$ 30 fps $\approx$ 22.7 Mbit/s. A configuration mode allows to set and read out the internal monitoring and control registers.

B. Warp Interpolation

The warp interpolation works on warps in *quadmesh* format, as this allows for a simple calculation of the Jacobian. As shown in Figure 3a), the two input warps are first upsampled using bilinear interpolation before being interpolated to the desired views. After the view interpolation, the quadmesh format is converted into a target coordinate and a Jacobian using finite differences. The term *halfquad* in the figure indicates that only half of a quad (i.e. two vertices) are computed at once (this is possible due to the scan-line operation). The output of this unit are packets (so called *splat jobs*) containing the view number, the Jacobian, the target coordinate and the associated input image pixel. All sub-units of the warp interpolation stage are matched in throughput if $N = 4$ dense warps have to be generated from each of the two input warps. In this case, this stage can deliver two splat jobs in each cycle.

C. Rendering

Each render unit in Figure 4 contains a kernel setup stage, which iteratively prepares the Gaussian filter kernels such that they can be efficiently evaluated in the rasterizer units. Only the required subpixels of the target image are evaluated, and the rasterizers are designed such that they can pre-calculate the indices of the next valid subpixels in the sparse interleaving pattern, see Figure 3b). Each render unit is able to process one splat job in four cycles, which translates into a throughput of 75 Msplat/s per second. This is sufficient to resample 1080p images at 30 fps as this amounts to 62.21 Msplat/s. As can be seen in Figure 5a), the average amount of subpixels that have to be rasterized per splat is around 12 when eight views are enabled. Since the rasterizers have a throughput of one subpixel per cycle, three rasterizers are allocated per render unit.

D. Accumulation

Although only the required subpixels are evaluated in the render units, the number of subpixels that need to be accumulated is still very large: $8 \times 1920 \times 1080 \times 30 \times 4 \approx 1.99$ Gsubpixel/s per color channel with eight enabled views. As a result, for each clock cycle, 6.63 subpixel values have to be accumulated per color channel. Fortunately, the large overlap among subsequent splats *of the same view* can be leveraged to reduce this number by placing small fully-associative subpixel-caches right after the rasterizers (so called *Level-1* caches in Figure 2). These L1 caches reduce the required accumulations by roughly a factor of 5.6 as shown in Figure 5b). The L2 cache is the actual framebuffer and is implemented as a sliding window which automatically adjusts its position depending on the incoming subpixel addresses. Assumptions on the geometric arrangement (i.e. almost rectified input images) of the views allow to store only a small excerpt (25 rows) of the whole output image on-chip, and therefore no external memory is required. Column interleaving with two memory banks per color channel is employed in order to provide sufficient bandwidth of 2 subpixels per cycle and color channel.

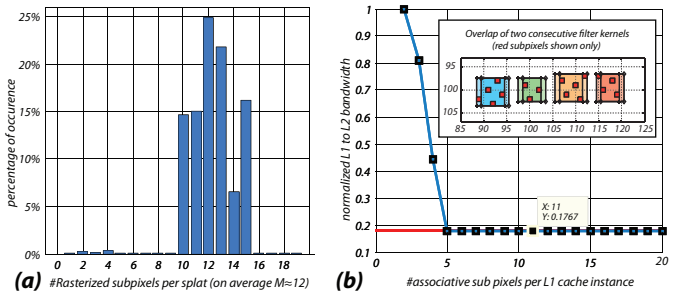

Fig. 5. a) On average, twelve subpixels have to be rasterized per splat. b) simulation that shows the L1 cache efficiency. In our implementation we use 6 associative subpixels per L1 cache instance.

TABLE I

KEY FIGURES OF THE MADMAX CHIP.

Physical Characteristics	
Technology / Package	UMC 65 nm, 8 M / CQFP 120
Core / Pad Voltage	1.2 V / 2.5 V
Core Area	14.2 mm^2
Circuit Complexity (incl. SRAM)	6.8 MGE (9.76 mm^2)
Logic (std. cells)	2.3 MGE (3.3 mm^2)
Warp- / L2 Buffer (SRAM)	0.76 MBit / 3.6 MBit
Functional Characteristics	
S3D Input- / Display Resolution	1920 × 1080 pixel
Max. #views / Max. y-Disparity	9 / 11.25 pixel
Max. Scrambling Pattern Size	10×10×3 subpixel
Performance (Core @1.2V, 8 Views)	
Max. Clock Frequency	260 MHz (Core), 65 MHz (I/O)
Max. Throughput	28.7 fps (8 views interleaved)
Power Dissipation	550 mW (Core), 350 mW (I/O)

TABLE II

COMPARISON WITH DIBR ARCHITECTURES.

	this work	**[3]**	**[5]**
Technology	UMC 65 nm	UMC 90 nm	TSMC 40 nm
Clock Frequency	260 MHz	200 MHz	240 MHz
Tot. Complexity	6.8 MGE	765.2 kGE	N/A
Logic	2.3 MGE	268.5 kGE	1.416 MGE
SRAM	3.6 MBit	554.4 kBit	159.2 kBit
Ext. Memory	no	yes	yes
Performance	28.7 fps	32.4 fps	216 fps
Resolution	1920 × 1080	1920 × 1080	4096 × 2160
Format	8 views interleaved	single frame	single frame
Video Decoder	-	-	H.264/AVC

IV. IMPLEMENTATION AND RESULTS

The implemented chip (Figure 6) is named MADMAX and was fabricated in a 65 nm CMOS technology. The key figures of the ASIC are shown in Table I. The term MGE stands for *mega gate equivalents*, and one pixel consists of three subpixels (RGB). The power dissipation has been measured on a Advantest SoC V93000 tester by looping the testvectors for the calculation of the first 300 rows of a frame with 8 enabled views (averaging over 10'000 DC measurement samples).

By switching into a configuration mode, many parameters such as the view positions, the scrambling pattern and the filter constants can be re-configured at runtime. The chip can also be used for other applications e.g. depth remapping [12] of stereo video with direct rendering into a column interleaved output image (with anisotropic anti-aliasing). For convenience during prototype testing, our design features a warp buffer able to store two whole warps (760 Kbit). The core of the chip runs at a four times faster clock than the I/Os). Both clocks have a fixed phase relationship and are generated on-chip from the same source clocks.

V. CONCLUSIONS

To our knowledge, this is the first implementation of IDW based MV rendering. The key innovations of our design are the analytically integrated display pre-filter, the fast, programmable rasterizers which are able to evaluate the splats at sparse sampling points, and the fully associative L1 caches which effectively exploit the spatial overlap of subsequent splats that are evaluated on a sparse grid only.

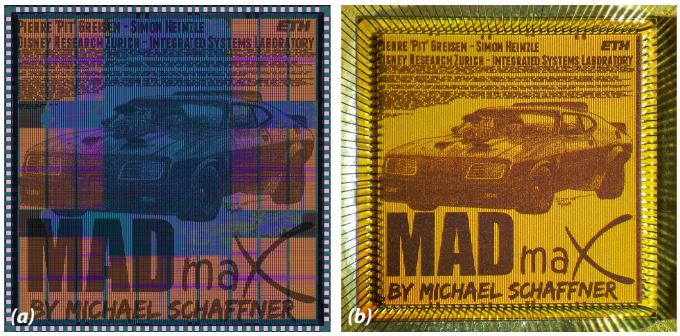

Fig. 6. CAD rendering (a) and photo (b) of the ASIC. The artwork is by Dan Poll (used with permission) and was placed between the power stripes.

The same architecture could easily support quad-full-HD (3840 × 2160) output resolution at 28.7 fps by increasing the I/O bandwidth of the L2 cache and only practical I/O limitations (no flip-chip) have prevented us from doing so. We believe that with further algorithmic (e.g. as proposed in [13]) and architectural optimizations, IDW based MV rendering systems can achieve comparable hardware complexity to mature DIBR implementations, such as [3]–[5] (a comparison is given in Table II).

REFERENCES

[1] J. Konrad and M. Halle, "3-D displays and signal processing," *Signal Processing Magazine, IEEE*, vol. 24, no. 6, pp. 97 –111, 2007.

[2] R.-P. M. Berretty, F. J. Peters, and G. T. G. Volleberg, "Real-time rendering for multiview autostereoscopic displays," *Proc. SPIE 6055, Stereoscopic Displays and Virtual Reality Systems XIII*, pp. 60 550N–60 550N–12, 2006.

[3] Y.-R. Horng, Y.-C. Tseng, and T.-S. Chang, "Vlsi architecture for real-time hd1080p view synthesis engine," *Circuits and Systems for Video Technology, IEEE Transactions on*, vol. 21, no. 9, pp. 1329 –1340, 2011.

[4] F.-J. Chang, Y.-C. Tseng, and T.-S. Chang, "A 94fps view synthesis engine for HD1080p video," in *Visual Communications and Image Processing (VCIP), 2011 IEEE*, 2011, pp. 1–4.

[5] P.-K. Tsung et al., "A 216fps 4096x2160p 3dtv set-top box soc for free-viewpoint 3dtv applications," in *Solid-State Circuits Conference Digest of Technical Papers (ISSCC), IEEE International*, 2011, pp. 124–126.

[6] M. Farre, O. Wang, M. Lang, N. Stefanoski, A. Hornung, and A. Smolic, "Automatic content creation for multiview autostereoscopic displays using image domain warping," in *Multimedia and Expo (ICME), IEEE International Conference on*, 2011, pp. 1–6.

[7] G. Wolberg, *Digital image warping*. IEEE Computer Society press, 1990, vol. 3.

[8] N. Stefanoski, M. Lang, and A. Smolic, "Image quality vs rate optimized coding of warps for view synthesis in 3d video applications," in *Image Processing (ICIP), 19th IEEE International Conference on*, 2012.

[9] P. Greisen, M. Schaffner, S. Heinzle, M. Runo, A. Smolic, A. Burg, H. Kaeslin, and M. Gross, "Analysis and VLSI implementation of EWA rendering for real-time HD video applications," *Circuits and Systems for Video Technology, IEEE Transactions on*, vol. 22, no. 11, pp. 1577 –1589, nov. 2012.

[10] P. Greisen, R. Emler, M. Schaffner, S. Heinzle, and F. Gurkaynak, "A general-transformation ewa view rendering engine for 1080p video in 130 nm cmos," in *VLSI and System-on-Chip (VLSI-SoC), 2012 IEEE/IFIP 20th International Conference on*, 2012, pp. 105–110.

[11] J. Konrad and P. Agniel, "Subsampling models and anti-alias filters for 3-d automultiscopic displays," *IEEE Transactions on Image Processing*, vol. 15, no. 1, pp. 128 –140, January 2006.

[12] M. Lang, A. Hornung, O. Wang, S. Poulakos, A. Smolic, and M. Gross, "Nonlinear disparity mapping for stereoscopic 3D," *ACM Trans. on Graphics (Proc. SIGGRAPH)*, vol. 29, no. 3, 2010.

[13] M. Schaffner, P. Greisen, S. Heinzle, and A. Smolic, "Efficient image resampling for multiview displays," in *Proceedings of the International Conference on Acoustics, Speech, and Signal Processing (ICASSP)*, 2013, to be published.