# Automatic Multiview Synthesis – Towards a Mobile System on a Chip

Michael Schaffner [1,2], Frank K. Gürkaynak [1], Hubert Kaeslin [1], Luca Benini [1,3], Aljosa Smolic [2]

[1] *Integrated Systems Laboratory, ETH Zürich, Switzerland*
[2] *Disney Research Zurich, Switzerland*
[3] *Università di Bologna, Italy*

{schaffner,kgf,kaeslin,benini}@iis.ee.ethz.ch, smolic@disneyresearch.com

***Abstract*—Over the last couple of years, multiview autostereoscopic displays (MADs) have become commercially available which enable a limited glasses-free 3D experience. The main problem of MADs is that they require several (typically 8 or 9) views, while most of the 3D video content is in stereoscopic 3D (S3D) today. In order to bridge this gap, the research community started to devise automatic multiview synthesis (MVS) methods. These algorithms require real-time processing and should be portable to end-user devices to develop their full potential. In this paper, we present a complete hardware system for fully automatic MVS. We give an overview of the algorithmic flow and corresponding hardware architecture, and provide implementation results of a hybrid FPGA/ASIC prototype – which is the first complete hardware system implementing image-domain-warping-based MVS. The proposed hardware IP could be used as a co-processor in a system-on-chip (SoC) targeting 3D TV sets, thereby enabling efficient content generation in real-time.**

## I. Introduction

Over the last couple of years, video capture, post-processing and distribution technologies for *stereoscopic 3D* (S3D) content have become mature enough for broad commercialization. Coupled with the box office success of S3D movies, this has brought on renewed interest in the development of S3D-capable consumer-electronic devices such as TV sets. However, most such devices require the viewers to wear some sort of shutter- or polarization glasses, which is often regarded as an inconvenience. Recently, so-called *multiview autostereoscopic displays* (MADs) [1], [2] have become commercially available. These are able to project several views of a scene simultaneously - enabling a glasses-free 3D experience and a limited motion parallax effect in horizontal direction. However, appropriate content for such displays is largely inexistent since storage and transmission of *high definition* (HD) content with more than two views is impractical and even infeasible in some cases. The fact that each MAD has different parameters exacerbates the problem. In order to bridge this *content-display gap*, *multiview synthesis* (MVS) methods [3], [4] have been devised over the past couple of years. These methods are able to generate $M$ virtual views from a small set of $N$ input views.

Common MVS methods are based on *depth image based rendering* (DIBR) [4], [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. Our work uses an alternative conversion concept suggested by [6] which is based on *image domain warping* (IDW). The IDW framework allows to locally distort image regions via a non-linear, two-dimensional transformation which is obtained by solving a least squares (LS) problem. The constraints for this problem are formulated using image features extracted from the input images.This technique is promising as it does not rely on pixel dense depth, but only on robust, sparse point correspondences. Further, no inpainting is required which is still an algorithmically difficult step of DIBR-based view synthesis methods [7].

Multiview synthesis, using IDW methods as well as alternative approaches, are computationally intensive - yet they should run efficiently in real-time and should be portable to end-user devices to develop their full potential. To this end, we develop an efficient hardware architecture for fully automatic MVS. The proposed architecture could be used as a co-processor in a *system-on-chip* (SoC) targeting 3D TV sets, thereby enabling efficient content generation in real-time. This paper[1] gives an overview of the algorithmic flow and the corresponding architecture, as well as implementation results of our hardware prototype – which is the first IDW-based real-time MVS system.

## II. Related Work

The IDW method used in this work relies heavily on prior work on warping approaches for video content adaption such as aspect-ratio retargeting [9], S3D retargeting [10] and non-linear disparity mapping [11]. In these applications, the video is warped in a content-aware manner in order to fulfill certain constraints. Image features (such as edges) and visual importance maps (saliency) are used in order to determine which parts of the image are important and should be preserved. Unavoidable distortions are moved to visually unimportant regions. This IDW framework has been extended for automatic MVS by [6]. The artifacts caused by geometrical incorrectness are minor and visually hardly noticeable. In fact, exhaustive and formal subjective experiments performed by MPEG [6] revealed that IDW as proposed here performs at least as well as DIBR-based [3], [5] methods, even if those used pre-computed and hand-tuned depth maps. Based on these results, MPEG adopted warp coding for the 3D extension of the new HEVC standard [12], thereby enabling IDW-based MVS.

[1] A longer, more detailed description of the system has been submitted to TCSVT for review [8].

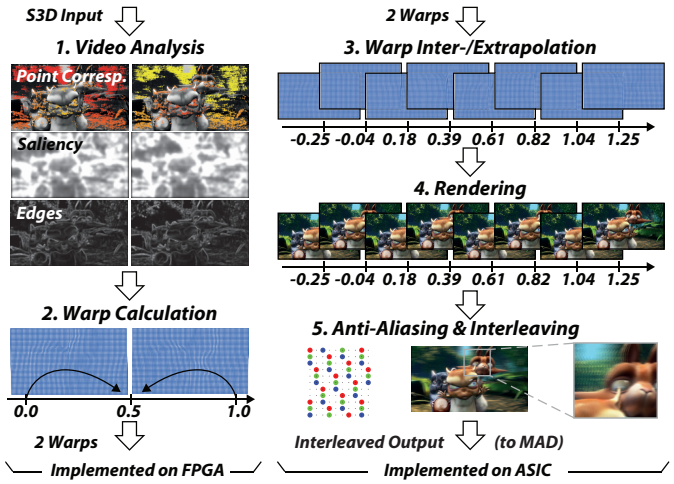


Fig. 1: Image domain warping pipeline for MVS (refer to the text for more information). As indicated below, the first part is implemented on an FPGA, and the second part on an ASIC. Testimages © copyright 2008, Blender Foundation.

Most published real-time systems either only implement the video analysis or the view synthesis modules. E.g., [13], [14] implement FPGA depth estimation cores and [15], [16] describe ASIC solutions for DIBR rendering. There are relatively few real-time systems which contain all modules, such as [17] and [18]. Riechert, *et al.* [17] presents the first complete system for DIBR-based multiview synthesis which is implemented using a high-end workstation, whereas Liao, *et al.* [18] describes an FPGA-based system able to synthesize one synthetic view from 1920×1080 S3D input at 60 fps from perfectly rectified video content.

In contrast, we implement a complete hardware system that only requires a stereo video input stream and that outputs eight interleaved views for display on a MAD.

## III. Algorithmic Flow

This section is a summary of the algorithmic flow of the implemented MVS scheme, and is based on [6], [19]. As shown in Figure 1, the input to the IDW processing pipeline is the S3D footage (left and right images) which is analyzed in order to reveal image features such as point correspondences, edges and saliency information in a first step. Those features are then used to formulate a global energy minimization problem. The solution of this problem results in 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, followed by resampling the S3D input according to those interpolated warps. Finally, the generated views are interleaved such that they can be displayed on the MAD. The individual steps are explained below.

### A. Video Analysis.

*1) Sparse Point Correspondences:* In multiview synthesis, the disparities are the most important features since they reveal the 3D geometry of the observed scene. Yet these have to be robust in order to get good results. As opposed to DIBR-based methods, the IDW approach used here works on sparse disparities. In this work we adopted features based on *semantic kernels binarized* (SKB) [20], since they work very well in the setting of almost ideally rectified stereo video. SKB features a low outlier rate and, therefore, it is possible to use the point correspondences without an additional RANSAC filtering step, which would be costly in hardware. Occasional outliers can be tolerated since the warp calculation process enforces spatial and temporal smoothness. Furthermore, since SKB is a binary descriptor, it can be calculated and matched very efficiently in hardware. Descriptors containing fixed-point or even floating-point entries such as SURF or SIFT are much more costly in this respect.

*2) Saliency and Edges:* A saliency map identifies the visually important regions in the image, and is used to guide the warp calculation such that deformations are hidden in unimportant regions (e.g. homogeneous parts such as blue sky). Extracting visual saliency is difficult since it is a subjective measure that depends to some extent on video content, viewer, and application. Here we use the *quaternion-Fourier-transform*-based (QFT) algorithm from [21] which exhibits a good tradeoff between computational complexity and quality. The algorithm leverages the *phase-spectrum* of a video sequence, which carries information on where discernible objects are located in an image. The QFT can be efficiently calculated using two separate 2D *Fast Fourier Transforms* (FFTs).

Very salient lines should also be preserved in order to avoid bending them in the warped images. This can be done using several methods, e.g. by extracting straight lines using the Hough transform. However, since this feature is not as important as the saliency and the point correspondences, simple gradient-magnitude maps without Canny edge post-processing are used here. Such gradient-magnitude maps can be extracted very efficiently using $3 \times 3$ Sobel filters.

### B. Warp Generation

*1) Energy Minimization:* The warps are calculated by solving a quadratic energy minimization problem of the form

$$\min_{\mathbf{f}} \left( E\left(\mathbf{f}\right)\right) = \min_{\mathbf{f}} \left( E_{data}\left(\mathbf{f}\right) + E_{smooth}\left(\mathbf{f}\right)\right) \qquad (1)$$

where the data term $E_{data}$ enforces function values at certain coordinate positions, and the smoothness term $E_{smooth}$ is a regularizer that propagates these known values to adjacent sampling positions. The vector $\mathbf{f}$ holds the samples of the unknown warp vertices. The data term itself is composed of the energy term $E_{pt}$ containing the desired vertex positions in the target image, and the energy term $E_t$ which enforces temporal consistency:

$$E_{data} = \lambda_{pt} E_{pt} + \lambda_t E_t.$$

The parameters $\lambda_{pt}$ and $\lambda_t$ are weighting parameters that can be used to set the relative importance of a particular constraint. Analogously, the smoothness term comprises the two constraints $E_{sal}$ and $E_{edge}$ that determine the local smoothing strength (also called *stiffness* sometimes):

$$E_{smooth} = \lambda_{sal} E_{sal} + \lambda_{edge} E_{edge}.$$

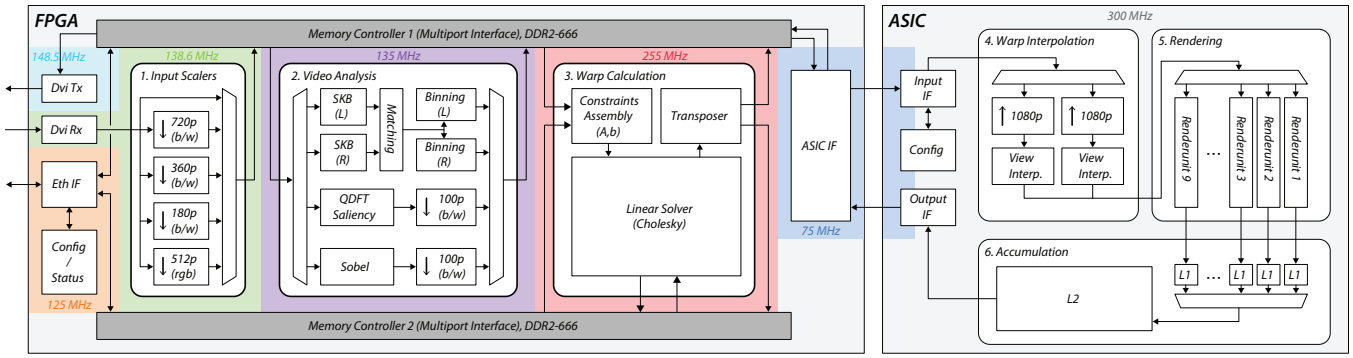


Fig. 2: Blockdiagram of the MVS prototype. It consists of five main parts, where the first three reside on the FPGA and the remaining ones on the ASIC.

Again, the parameters $\lambda_{sal}$ and $\lambda_{edge}$ are relative importance weights. Recall that we solve for two warps, each of which maps either the left or the right image to a virtual view at position $0.5$ on the normalized baseline (see Figure 1). This asks for setting up four quadratic energy functionals $E^{lx}$, $E^{ly}$, $E^{rx}$ and $E^{ry}$ (two coordinate dimensions for each input view) which can all be minimized independently. More details on how the constraints are formulated can be found in [6], [19].

*2) Sparse Linear System:* Since the energy functionals to be minimized are quadratic in the elements of $\mathbf{f}$, the solution of the problem in (1) is a *least-squares* (LS) solution. The constraints are defined on small neighbourhoods, and therefore the corresponding linear system is very sparse and only contains one main- and nine off-diagonals. Further, the matrix dimension is in the order of tens of thousands to millions - depending on the resolution of the warping grid. The resolution of current video content is predominantly $1920\times1080$, resulting in four equation systems ($E^{lx}$,$E^{ly}$,$E^{rx}$ and $E^{ry}$) with nearly two million variables for each frame in the video. Solving such large systems at frame rates of up to 30 fps is computationally very demanding and infeasible for embedded real-time applications. Also, such pixel-dense solutions are not necessary as even $10\times$ sub-sampled grids result in sufficient synthesis quality on many S3D HD sequences [6]. This results in realistic grid sizes of about $180\times100$, which corresponds to 18 k variables. Based the evaluations by [19], we decided to use a Cholesky-decomposition-based solver in our hardware implementation, since it provides a good tradeoff between memory bandwidth and hardware complexity in this setting.

### C. Warp Interpolation and Rendering

In the rendering step, the two warps are first bilinearly upsampled in order to form pixel-dense warps. Then, these two warps are linearly inter- and extrapolated to all desired view positions on the normalized baseline. As illustrated in Figure 1, eight views are generated in this setting: four views are from the left image, and the other four views are generated from the right image. Note that this view interpolation can be used to linearly scale the depth range of the scene at runtime. The input images are then resampled using *elliptical weighted average* (EWA) splatting [22], which is a *forward-mapping* method. Although EWA is more complex than traditional bilinear backward mapping, it has the advantage that no warp inversion is required as the warps are generated in forward format in this application. The EWA framework uses Gaussian filter kernels and local, affine approximations of the image warp in order to calculate the footprint of an input image pixel in the output image. 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 [23].

## IV. Hardware Architecture

Figure 2 provides a high-level block diagram of our MVS system and can be conceptually divided into two parts: the core view synthesis components (numbered from 1. through 6.) which are, in principle, device-independent and can be ported to other FPGA/ASIC technologies; and an infrastructure part handling all FPGA board and I/O specific controllers. The core components 1. to 6. process the input video according to the algorithmic flow presented in Section III: 1. The input scaler block scales the input video to the different resolutions that are required later on, and stores them in the memory. 2. The video analysis block extracts the saliency- and edge maps, and calculates the point correspondences using these scaled video frames. 3. The warp calculation block contains a constraints assembly core and a Cholesky-decomposition-based linear solver. The former is a microcode-programmable unit that builds the LS-problem matrices, and the latter is a fixed function solver. 4. The warp interpolation stage upsamples the two warps to 1080p resolution, and interpolates them to the desired view positions. The rendering block 5. prepares a filter kernel for each input pixel/warp-vertex pair, evaluates it on the sampling lattice of the assigned view on the display, and sends it to the accumulation stage 6. which fuses all subpixels to form the interleaved output image.

## V. Implementation Results

Table I provides a summary of the resource utilization of the multiview prototype (shown in Figure 3). The system has been designed to provide enough throughput to process 1080p S3D input footage @30 Hz using warps with $180 \times 100$ vertices. Note that the same rendering architecture implemented in the ASIC could easily support UHD ($3840 \times 2160$) output

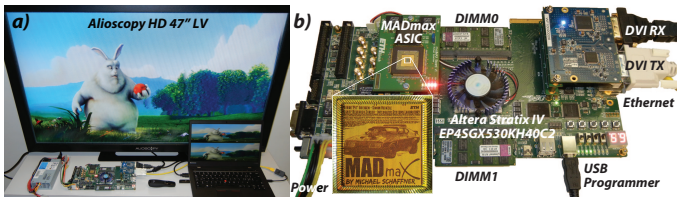


Fig. 3: a) Photograph of the running system, and b) close-up of the hardware prototype. Testimage © copyright 2008, Blender Foundation.

resolution at the same frame rate by increasing the I/O bandwidth. This has not been carried out here due to practical I/O limitations (no flip-chip packaging). The only other systems with similar functionality are the works by [17] and [18]. [17] implement a system based on a dual processor workstation with two Intel Xeon 5690 CPU's and two NVIDIA GTX 590 graphics cards. It has almost the same performance as ours and is able to synthesize 8 interleaved views for an Alioscopy display from 1080p S3D content at 24 fps. Clearly, the size of this system, the cost, and power consumption make it unsuitable for integration into consumer devices. [18] implement a single-view synthesis pipeline on a FPGA able to produce depth-adjusted S3D content generated from 1080p S3D input video at 60 fps. In order to do so, they synthesize one virtual view and bypass one of the input views. However, their implementation is not a complete MVS system since it only renders one virtual view and not up to nine views as in our work and [17]. [18] also make the assumption that the input video is perfectly rectified, i.e. there is no $y$-disparity between the left and right image, which simplifies all 2D-problems to 1D-problems. Yet such rectification is not easily achieved, and therefore systems should be designed to tolerate y-disparities. For example, our system is designed to support up to $\pm 11$ pixels of y-disparity.

## VI. Conclusions

To our knowledge, this is the first hardware implementation of a complete, IDW-based MVS system, capable of full HD view synthesis with 8 views in real-time. During informal subjective tests using full-length S3D movie footage, we found that the hardware system works well on a broad range of content using the same set of parameters. The current hardware prototype is based on an FPGA platform, combined with a custom rendering ASIC. The developed hardware IP could be integrated within a SoC where it would act as an energy-efficient co-processor – thereby paving the way towards mobile MVS in real-time.

TABLE I: Physical characteristics of the different parts of the multiview system. The FPGA is an Altera Stratix IV (EP4SGX530KH40C2), and the ASIC has been fabricated in 65 nm technology from UMC.

| **FPGA** | Logic | | | RAM | | Freq. |
|---|---|---|---|---|---|---|
| | LUTs | Regs | DSPs | LUTs | 9/144K | [MHz] |
| Scalers | 4.1 k | 7.1 k | 24 | 2.3 k | 10/0 | 138.6 |
| Analysis | 43.3 k | 47.3 k | 177 | 2.7 k | 434/24 | 135 |
| Warp Calc. | 51.9 k | 67.4 k | 420 | 0.7 k | 148/0 | 255 |
| Subtotal | 99.3 k | 121.8 k | 621 | 5.7 k | 592/24 | ↑ |
| IO/Infra. | 22.1 k | 35.5 k | 0 | 15.7 k | 14/4 | N/A |
| Total | **121.4 k** | **157.3 k** | **621** | **21.4 k** | **606/28** | ↑ |
| **ASIC** | Logic | | | SRAM | | Freq. |
| | [mm$^2$] | [kGE] | [MBit] | [mm$^2$] | [kGE] | [MHz] |
| IO/Infra. | 0.28 | 195 | - | - | - | 75, 300 |
| Warp Interp. | 0.314 | 218 | 0.76 | 1.023 | 710 | 300 |
| Rendering | 2.317 | 1'609 | - | - | - | 300 |
| Accumulator | 0.385 | 267 | 3.6 | 5.436 | 3'775 | 300 |
| Total | **3.296** | **2'289** | **4.36** | **6.459** | **4'485** | ↑ |

## References


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

[2] A. Boev, R. Bregovic, and A. Gotchev, "Signal Processing for Stereoscopic and Multi-View 3D Displays," in *Handbook of Signal Processing Systems*. Springer New York, 2013, pp. 3–47.

[3] D. Tian, P. Lai, P. Lopez, and C. Gomila, "View Synthesis Techniques for 3D Video," *Proc. SPIE*, vol. 7443, pp. 74 430T–74 430T–11, 2009.

[4] C. Zhu, Y. Zhao, L. Yu, and M. Tanimoto, *3d-tv System with Depth-image-based Rendering*. Springer, 2014.

[5] M. Tanimoto, T. Fujii, and K. Suzuki, "View Synthesis Algorithm in View Synthesis Reference Software 3.5 (VSRS3. 5) Document M16090, ISO/IEC JTC1/SC29/WG11 (MPEG)," 2009.

[6] N. Stefanoski, O. Wang, M. Lang, P. Greisen, S. Heinzle, and A. Smolic, "Automatic View Synthesis by Image-Domain-Warping," *IEEE TIP*, vol. 22, no. 9, pp. 3329–3341, 2013.

[7] P. Ndjiki-Nya, M. Köppel, and D. Doshkov et. al., "Depth Image Based Rendering with Advanced Texture Synthesis," in *IEEE ICME*, July 2010.

[8] M. Schaffner, F. K. Gürkaynak, P. Greisen, H. Kaeslin, L. Benini, and A. Smolic, "Hybrid ASIC/FPGA System for Fully Automatic Stereo-to-Multiview Conversion using IDW," 2015, submitted to TCSVT.

[9] M. Rubinstein, D. Gutierrez, O. Sorkine, and A. Shamir, "A Comparative Study of Image Retargeting," in *ACM TOG*, vol. 29, no. 6, 2010, p. 160.

[10] S. Kopf, B. Guthier, C. Hipp, J. Kiess, and W. Effelsberg, "Warping-Based Video Retargeting for Stereoscopic Video," *IEEE ICIP*, 2014.

[11] M. Lang, A. Hornung, O. Wang, S. Poulakos, A. Smolic, and M. Gross, "Nonlinear Disparity Mapping for Stereoscopic 3D," *ACM TOG*, vol. 29, no. 4, p. 75, 2010.

[12] N. Stefanoski, M. Lang, and A. Smolic, "Image Quality vs. Rate Optimized Coding of Warps for View Synthesis in 3D Video Applications," in *IEEE ICIP*, Sept 2012, pp. 1289–1292.

[13] M. Werner, B. Stabernack, and C. Riechert, "Hardware Implementation of a Full HD Real-Time Disparity Estimation Algorithm," *IEEE TCE*, vol. 60, no. 1, pp. 66–73, February 2014.

[14] A. Akın, I. Baz, A. Schmid, and Y. Leblebici, "Dynamically Adaptive Real-Time Disparity Estimation Hardware using Iterative Refinement," *Integration, the VLSI Journal*, vol. 47, no. 3, pp. 365 – 376, 2014.

[15] Y. Horng, Y. Tseng, and T. Chang, "Vlsi architecture for real-time hd1080p view synthesis engine," *IEEE TCSVT*, vol. 21, no. 9, 2011.

[16] P.-K. Tsung et al., "A 216fps 4096x2160p 3DTV Set-Top Box SoC for Free-Viewpoint 3DTV Applications," in *IEEE ISSCC*, 2011.

[17] C. Riechert, F. Zilly, P. Kauff, J. Güther, and R. Schäfer, "Fully Automatic Stereo-to-Multiview Conversion in Autostereoscopic Displays," *The best of IET and IBC*, vol. 4, no. 8, p. 14, 2012.

[18] C. Liao, H. Yeh, K. Zhang, V. Geert, T. Chang, and G. Lafruit, "Stereo Matching and Viewpoint Synthesis FPGA Implementation," in *3D-TV System with Depth-Image-Based Rendering*. Springer New York, 2013.

[19] P. Greisen, M. Runo, P. Guillet, S. Heinzle, A. Smolic, H. Kaeslin, and M. Gross, "Evaluation and FPGA Implementation of Sparse Linear Solvers for Video Processing Applications," *IEEE TCSVT*, vol. 23, no. 8, pp. 1402–1407, 2013.

[20] F. Zilly, C. Riechert, P. Eisert, and P. Kauff, "Semantic Kernels Binarized - A Feature Descriptor for Fast and Robust Matching," in *CVMP*, nov. 2011, pp. 39 –48.

[21] C. Guo, Q. Ma, and L. Zhang, "Spatio-Temporal Saliency Detection Using Phase Spectrum of Quaternion Fourier Transform," in *IEEE CVPR*, June 2008, pp. 1–8.

[22] M. Zwicker, H. Pfister, J. V. Baar, and M. Gross, "EWA Splatting," *IEEE TVCG*, vol. 8, no. 3, pp. 223–238, 2002.

[23] M. Schaffner, P. Greisen, S. Heinzle, F. K. Gurkaynak, H. Kaeslin, and A. Smolic, "MADmax: A 1080p Stereo-to-Multiview Rendering ASIC in 65 nm CMOS based on Image Domain Warping," in *Proceedings of ESSCIRC*, 2013, pp. 61–64.