Analysis and VLSI Implementation of EWA Rendering for Real-time HD Video Applications

Pierre Greisen*†, Michael Schaffner*, Simon Heinzle†, Marian Runo*,
Aljosa Smolic†, Andreas Burg‡, Hubert Kaeslin*, Markus Gross*†
*ETH Zurich, †Disney Research Zurich, ‡EPF Lausanne

***Abstract*—Non-linear image warping or image resampling is a necessary step in many current and upcoming video applications such as video retargeting, stereoscopic 3D mapping, and multi-view synthesis. The challenges for real-time resampling include image quality but also available energy and computational power of the employed device. In this work, we employ an elliptical-weighted average (EWA) rendering approach to 2D image resampling. We extend the classical EWA framework for increased visual quality and provide a VLSI architecture for efficient view rendering. The resulting architecture is able to render high-quality video sequences in real-time targeted for low-power applications in end-user display devices.**

***Index Terms*—rendering, EWA splatting, image-based rendering, VLSI, video processing**

I. Introduction

Visual communication has become ubiquitous. Today, we consume visual content on a broad range of displays, from large scale cinema screens, television sets, and personal computer screens to various types of mobile devices. Pixel resolution, aspect ratio, and frame rate of corresponding displays vary significantly. Also, capabilities of terminal devices greatly differ in terms of computational power, memory, and battery lifetime. Furthermore, the delivery of visual content is carried out over a large range of communication channels and protocols. To cope with the resulting heterogeneous environment in visual communication, scalable video coding (SVC) techniques efficiently represent and encode the same video content in different formats [1]. Channels and terminals may pick the right bits from the scalable stream to adapt to given capabilities and conditions. However, SVC still couples content creation to consumption and does not handle all possible cases.

The desired de-coupling can be achieved if the terminal device is able to *render* video in the desired display format. In this context, content-aware video retargeting recently received a lot of attention [2]: to change the aspect ratio of a video, the frames are transformed in a non-linear fashion, such that visually important regions keep their aspect ratio, while distortions are hidden in visually less important regions (see Fig. 1). High quality non-linear image warping (rendering) in the terminal device is a crucial component in such processing.

Further, the advent of stereoscopic 3D (S3D) for home entertainment and mobile applications creates new challenges

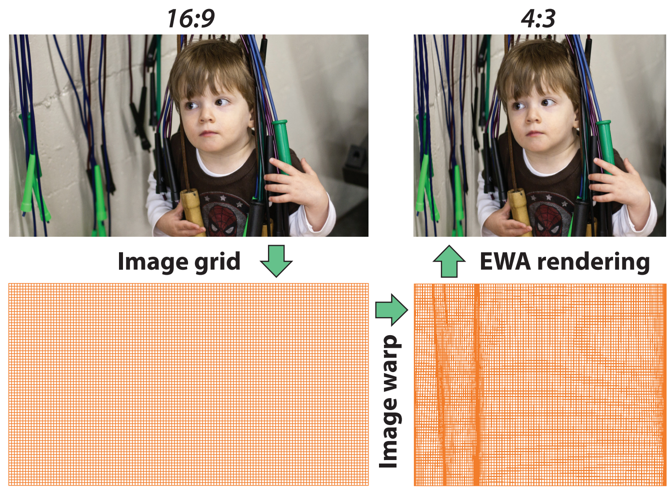


Fig. 1: *Example of EWA rendering used for aspect ratio retargeting [2]. Refer to Fig. 17 for details. (Image courtesy of Andrew Malone).*

for end-user devices in terms of rendering and view synthesis [3]. Depth impression in S3D is a sensitive illusion that largely depends on the display size and viewing distance. Disparity mapping allows for (non-linearly) adapting the depth impression of S3D content based on viewing conditions or user preferences [4], [5]. This enables for instance a depth button on the remote control of a 3DTV set, similar to brightness or color controls today. Also, disparity mapping requires view synthesis, which can be realized by non-linear image warping.

Finally, next generation visual communication applications will require even more sophisticated forms of view synthesis and rendering [3]. Multiuser autostereoscopic displays require a multiview signal as input, which can be generated from S3D for instance by non-linear image warping [6]. Free viewpoint video applications allow the user to select his own viewpoint and direction, which requires synthesis of the corresponding view [7], [8]. This may be embedded into a teleimmersion or telepresence application [9].

In consequence, all these advanced 2D and 3D video processing applications mentioned so far require non-linear image warping. Most of such processing is realized today on graphics processing units (GPUs), which are the natural choice for rendering applications [10]. Although rendering on GPUs achieves high performance, GPUs consume several 100 Watts. Also, GPUs are neither cheap nor small in size and hence ill-suited for many end-user devices such as smart phones or televisions. Further, the recently appearing mobile GPUs trade computational power for energy efficiency but always remain less efficient than custom architectures due to the programmability overhead. In this paper, we therefore present a custom hardware architecture to replace GPUs in the context of non-linear image warping, similar to work presented in [11] and [12] for point rendering. Our design enables the above-mentioned advanced applications at low cost and low power.

Algorithms for non-linear image warping, *resampling* or texture mapping have been extensively covered in literature [13], [14]. Among those algorithms, elliptical weighted average (EWA) filters provide a good tradeoff between visual quality and computational complexity, especially for non-linear transformations ([15], [16], [10]). Iterative methods such as [15] or [17] can provide even better quality, but they also involve much higher computational complexity and are less well suited for hardware implementation. The often-used bilinear filtering, for which also VLSI implementations have been proposed (such as in [18]), provides fair quality at low computational complexity for linear transformations, but can lead to poor results for non-linear transformations. We therefore select the EWA splatting algorithm [19] as starting point of our work. The main algorithmic drawback of EWA filters, i.e., over-blurring, can be extenuated by careful adjustment of the filter parameters, which is addressed in this work. We also show that EWA splatting can be efficiently implemented in VLSI in contrast to iterative high-quality methods or computationally intensive supersampling techniques.

a) Contributions: This work consists of two parts: an analysis and optimization of the EWA splatting *algorithm* and a corresponding VLSI *architecture* for real-time non-linear warping. First, we extend the traditional EWA splatting *algorithm* by showing how to optimally chose the filter parameters and by providing an adaptive scheme that optimizes the tradeoff between blurring and aliasing. Also, to practically deal with the infinite impulse response (IIR) of EWA filters, we show how to select cut-off points in the rendered target space. Secondly, to provide a low-power, low-cost, and small size solution, we propose a VLSI *architecture* of the derived EWA splatting algorithm for real-time, high-resolution non-linear warping. To cope with the large memory bandwidth requirements of EWA splatting, we propose a two-level caching architecture that significantly reduces the required memory bandwidth. Further, we investigate various number formats for EWA splatting. Finally, we provide area and performance results for a fabricated design in a 180 nm CMOS process.

b) Outline: The remainder of the paper is structured as follows: Sec. II reviews the basics of image resampling and EWA splatting in particular. In Sec. III we derive and discuss the optimum EWA filter parameters. Sec. IV summarizes the data flow of the implemented EWA splatting design and the assumptions made for the VLSI architecture. Sec. V explains the hardware details of the EWA splatting, with a particular emphasis on arithmetic precision and the proposed caching architecture. Sec. VI provides rendering quality results as well as ASIC performance and complexity results.

II. Background: Image Rendering and EWA Splatting

In this section, the necessary basics of image-based EWA rendering are summarized, based on [19] and references therein. In Sec. III we show how to set the parameters of the rendering formulas to maximize the rendering quality.

A. Notation

The following notation conventions and symbols are used throughout the paper. Scalars are represented by lower case letters, column-vectors by bold-face lower case letters, and matrices by uppercase letters. The entry in the ith row and jth column of a matrix A is denoted as $a_{i,j}$. The continuous convolution is denoted by a $*$ symbol. The Dirac-delta distribution is denoted by $\delta(\mathbf{x})$, with $\int \delta(\mathbf{x}) f(\mathbf{x}) d\mathbf{x} = f(0)$. The L_2 norm of a square integrable function $f(x)$ is denoted and defined as $||f(x)||_2 = \int_{D_f} |f(x)|^2 dx$, where D_f is the domain of $f(x)$. $|A|$ denotes the determinant of A.

B. Rendering

Given a 2D source image and a transformation function assigning a target coordinate to each source coordinate, a target image is *rendered* by mapping each source pixel position into a target pixel position and subsequently resampling the pixel values on an integer grid.

Let $\mathbf{u}_k \in \mathbb{N}^2$ be the kth discrete pixel position with intensity w_k in a uniformly sampled source image. The source image grid $\mathbf{u}_k$ is an integer pixel grid with finite dimensions. The domain of the source image index k is denoted as $D_\mathrm{s} = \{1, \dots, W_\mathrm{s} H_\mathrm{s}\}$ with image width W_s and image height H_s. The rendering process transforms an arbitrary pixel location $\mathbf{u}$ in the source image into a target pixel position $\mathbf{x}$

$$\mathbf{x} = m(\mathbf{u}),$$

where m is an arbitrary mapping (see Fig. 2). Assuming an integer grid for the source pixel positions $\mathbf{u}$, the transformed positions will generally not form an integer grid. Therefore, we introduce the continuous source image

$$\begin{aligned} f_\mathrm{s}(\mathbf{u}) &= \sum_{k \in D_\mathrm{s}} w_k \delta(\mathbf{u} - \mathbf{u}_k) * f_\mathrm{i}(\mathbf{u}), \\ &= \sum_{k \in D_\mathrm{s}} w_k \int_{\mathbb{R}^2} \delta(\boldsymbol{\tau} - \mathbf{u}_k) f_\mathrm{i}(\mathbf{u} - \boldsymbol{\tau}) d\boldsymbol{\tau}, \\ &= \sum_{k \in D_\mathrm{s}} w_k f_\mathrm{i}(\mathbf{u} - \mathbf{u}_k) \end{aligned}$$

where $f_\mathrm{i}(\mathbf{u})$ is a 2D interpolation function. Using the continuous source image and the pixel transformation mapping, the target image is

$$f_\mathrm{c}(\mathbf{x}) = \sum_{k \in D_\mathrm{s}} w_k f_\mathrm{i}(m^{-1}(\mathbf{x}) - \mathbf{u}_k).$$

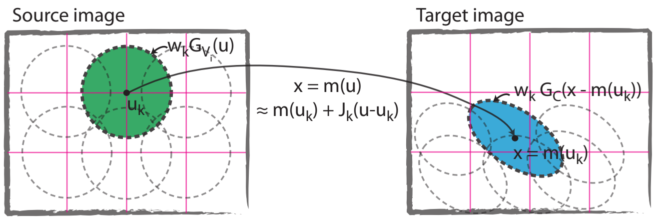


Fig. 2: *EWA Splatting or EWA target space rendering. Each source pixel is mapped into a screen pixel according to its spatially and temporally varying affine warp function J_k.*

In order to be displayed, $f_{\mathrm{c}}(\mathbf{x})$ must be sampled on a uniform integer grid. To avoid aliasing during this sampling step, an anti-aliasing filter $h(\mathbf{x})$ is applied

$$\begin{aligned} f_{\mathrm{c,aa}}(\mathbf{x}) &= f_{\mathrm{c}}(\mathbf{x}) * h(\mathbf{x}) \\ &= \int_{\mathbb{R}^2} \sum_{k \in D_{\mathrm{s}}} w_k f_{\mathrm{i}}(m^{-1}(\boldsymbol{\tau}) - \mathbf{u}_k) h(\mathbf{x} - \boldsymbol{\tau}) d\boldsymbol{\tau}. \end{aligned} \tag{1}$$

Eq. (1) provides the rendered target image for arbitrary interpolation filters, anti-aliasing filters, and mappings. Thus, generating an output image can be done by evaluating $f_{\mathrm{c,aa}}(\mathbf{x})$ on a 2D integer grid. To make the evaluation computationally tractable in real-time, an approximation to the mapping function and a specific set of filters are discussed in the following.

C. Linearization

The general mapping function $m(\mathbf{u})$ can be linearly approximated with a Taylor expansion around an integer grid position $\mathbf{u}_k$

$$\mathbf{x} = m(\mathbf{u}) \approx m(\mathbf{u}_k) + J_k \cdot (\mathbf{u} - \mathbf{u}_k),$$

where J_k is the 2×2 Jacobian matrix of m at position $\mathbf{u}_k$. The approximation error is small if the interpolation function has compact support around $\mathbf{u}_k$ (e.g., Gaussians) since the approximation is most precise in the vicinity of $\mathbf{u}_k$. Rearranging the expression into

$$\mathbf{u} = m^{-1}(\mathbf{x}) \approx J_k^{-1} \cdot (\mathbf{x} - m(\mathbf{u}_k)) + \mathbf{u}_k,$$

and substituting the approximation into (1) yields

$$\tilde{f}_{\mathrm{c,aa}}(\mathbf{x}) = \int_{\mathbb{R}^2} \sum_{k \in D_{\mathrm{s}}} w_k f_{\mathrm{i}}(J_k^{-1}\boldsymbol{\tau}) h(\mathbf{x} - m(\mathbf{u}_k) - \boldsymbol{\tau}) d\boldsymbol{\tau}. \tag{2}$$

D. EWA Splatting

Elliptical weighted average (EWA) splatting employs multi-dimensional elliptical Gaussian filters. For a covariance matrix V a Gaussian filter is defined as

$$G_V(\mathbf{x}) := \frac{1}{2\pi |V|^{1/2}} e^{-1/2 \mathbf{x}^T V^{-1} \mathbf{x}}. \tag{3}$$

Thus, in the EWA splatting setup, the anti-aliasing filter is a 2D Gaussian while the transformed interpolation filter is a Gaussian under an affine transformation

$$\begin{aligned} h(\mathbf{x}) &= G_{V_{\mathrm{a}}}(\mathbf{x}), \\ f_{\mathrm{i}}(J_k^{-1}\mathbf{x}) &= G_{V_{\mathrm{i}}}(J_k^{-1}\mathbf{x}) = \frac{1}{|J_k^{-1}|} G_{J_k V_{\mathrm{i}} J_k^T}(\mathbf{x}), \end{aligned}$$

where $V_{\mathrm{i}} = \mathrm{diag}(\sigma_{\mathrm{i},x}^2, \sigma_{\mathrm{i},y}^2)$ and $V_{\mathrm{a}} = \mathrm{diag}(\sigma_{\mathrm{a},x}^2, \sigma_{\mathrm{a},y}^2)$ are the diagonal interpolation and anti-aliasing covariance matrices, respectively. $\sigma_{\mathrm{i},x}^2$ is the interpolation variance in horizontal direction, σ_{i}^2 is the variance of an isotropic covariance matrix: $V_{\mathrm{i}} = \sigma_{\mathrm{i}}^2 I_2$. Substituting the Gaussian filters into (2) yields the EWA rendering or EWA splatting equation in target space

$$\begin{aligned} f_{\mathrm{EWA}}(\mathbf{x}) &= \sum_{k \in D_{\mathrm{s}}} w_k \frac{1}{|J_k^{-1}|} G_{J_k V_{\mathrm{i}} J_k^T + V_{\mathrm{a}}}(\mathbf{x} - m(\mathbf{u}_k)), \qquad (4) \\ &= \sum_{k \in D_{\mathrm{s}}} \frac{w_k |J_k|}{2\pi |C|^{1/2}} e^{-1/2 (\mathbf{x} - m(\mathbf{u}_k))^T C^{-1} (\mathbf{x} - m(\mathbf{u}_k))}. \end{aligned}$$

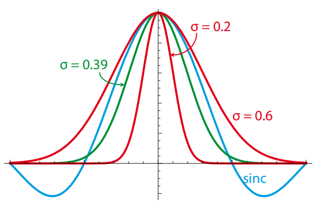
To obtain (4) we use the fact that a convolution of two Gaussians is again a Gaussian. The location index k of the EWA covariance matrix $C := J_k V_{\mathrm{i}} J_k^T + V_{\mathrm{a}}$ is omitted for ease of notation. Fig. 2 summarizes the EWA rendering process.

III. EWA Filter Parametrization

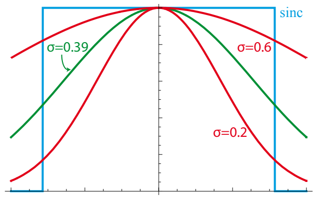
In order to achieve high-quality video rendering results, an optimal filter parameterization for the general EWA rendering equation (4) is crucial. In this section, we derive the optimal Gaussian interpolation covariance matrix V_{i} and develop a strategy to adaptively chose the anti-aliasing covariance matrix V_{a} to optimize the tradeoff between aliasing and blurring. Also, we derive cut-off points to truncate the filter support, denoted as *bounding box* of the Gaussian ellipse. The evaluation of the filters can thus be delimited to the significant contributions and the summation term is reduced to a small sampling region.

A. Interpolation and anti-aliasing parametrization

The Gaussian filter can introduce excessive blurring for large variances and can lead to aliasing for small variances. To achieve the best possible image rendering quality, we therefore derive the *optimal* trade-off between blurring and anti-aliasing. We first determine the optimal covariance matrix for the circular Gaussian interpolation filter in (uniformly sampled) source space $f_{\mathrm{i,EWA}}(\mathbf{x}) = G_{V_{\mathrm{i}}}(\mathbf{x})$. From this result, the optimal parameterization of the transformed interpolation kernel in target space $f_{\mathrm{i,EWA}}(J_k^{-1}\mathbf{x})$ follows immediately. The parametrization of the anti-aliasing filter by itself reuses the same values as determined for the interpolation filter in source space. However, due to the convolution with the target space interpolation filter, the resulting resampling filter is locally adaptive. Hence, an optimal parametrization requires an adaptive anti-aliasing strategy, which we propose in the subsequent section.



(a) Time domain



(b) Frequency domain

Fig. 3: *(Normalized) impulse responses of ideal (sinc) and Gaussian low-pass filters with different interpolation variances.*

1) Interpolation in source space: To find a good tradeoff for the interpolation covariance matrix V_{i}, we minimize the mean squared error (MSE) between the EWA filter and an ideal low pass filter. The ideal low-pass filter is a 2D sinc function $f_{\mathrm{i,ideal}}(\mathbf{x}) = \mathrm{sinc}(x)\mathrm{sinc}(y)$ which corresponds to a 2D rectangular function in frequency domain

$$\mathcal{F}f_{\mathrm{i,ideal}}(\mathbf{x}) =: \hat{f}_{\mathrm{i,ideal}}(\mathbf{p}) = \frac{1}{2\pi}\mathrm{rect}_{\pi}(p)\mathrm{rect}_{\pi}(q)$$

where $\mathbf{p} = (p,q)^T$ is a point in 2D angular frequency space and $\mathrm{rect}_{\pi}(p) = 1$ if $|p| \leq \pi$ and 0 else; $\mathcal{F}$ is the Fourier transform operator [20].

The Fourier transform of the EWA interpolation filter in source space is

$$\hat{f}_{\mathrm{i,EWA}}(\mathbf{p}) = \frac{1}{2\pi}\exp\left(-\frac{\sigma_i^2(p^2+q^2)}{2}\right),$$

where $V_{\mathrm{i}} = \sigma_{\mathrm{i}}^2 I_2$, and σ_{i}^2 is the interpolation variance. Note that the optimal source space covariance matrix V_{i} is isotropic, as the sampling in source space is assumed to be uniform.

In order to compare the EWA kernel and the ideal sinc kernel, we calculate their mean squared error (MSE)

$$\begin{aligned}
\mathrm{mse}(\sigma_i) &= ||f_{\mathrm{i,ideal}}(\mathbf{x}) - f_{\mathrm{i,EWA}}(\mathbf{x})||_2, \\
&= ||\hat{f}_{\mathrm{i,ideal}}(\mathbf{p}) - \hat{f}_{\mathrm{i,EWA}}(\mathbf{p})||_2, \\
&\propto ||\mathrm{rect}_{\pi}(p)\mathrm{rect}_{\pi}(q) - \exp(-\sigma_{\mathrm{i}}^2/2(p^2+q^2))||_2, \\
&\propto 1 + \frac{1}{4\pi\sigma_{\mathrm{i}}^2}\left(1 - 4\cdot\mathrm{erf}\left(\frac{\pi\sigma_{\mathrm{i}}}{\sqrt(2)}\right)\right),
\end{aligned} \tag{5}$$

where the first step follows directly from Parseval's theorem, and where $\mathrm{erf}(x)$ is the Gaussian error function. The best (least-squares) tradeoff between anti-aliasing and blurring can be obtained by choosing an interpolation variance such that the $\mathrm{mse}(\sigma_i)$ is minimized. Numerical minimization of (5) yields the optimal tradeoff in the least squares sense:

$$\hat{\sigma}_{\mathrm{i}} = \mathrm{argmin}_{\sigma_{\mathrm{i}}}(\mathrm{mse}(\sigma_{\mathrm{i}})) \approx 0.39.$$

A comparison of this ideal EWA low-pass filter to other EWA filters is plotted in Fig. 3. Note that, using the (ideal) sinc directly is not optimal in practice due to its slow decay and hence large support. A necessary truncation (due to complexity constraints) would lead to severe filter quality degradations (e.g., Gibbs oscillations).

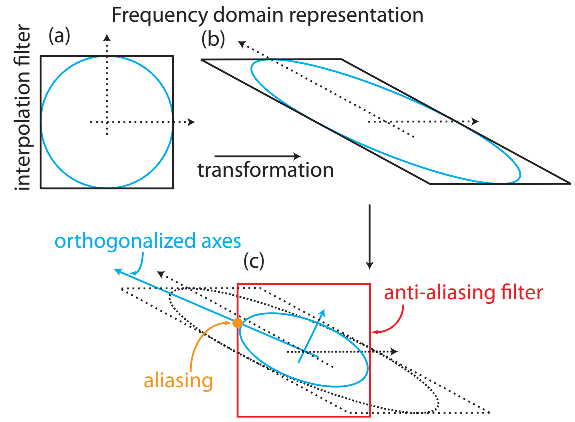


Fig. 4: *Different steps of EWA splatting in frequency domain. The transformed axes are not necessary the principal axes of the ellipse in target space; therefore, the transformation is diagonalized into an orthonormal basis. With the main axes an adaptive anti-aliasing rule is derived, which scales the axes if aliasing is detected in a specific direction.*

2) Interpolation in target space: In the EWA splatting case, we are not interested in the source space parametrization but in the target space parametrization. The target space parameterization $f_{\mathrm{i,EWA}}(J_k^{-1}\mathbf{x})$ can be directly derived from the source space parameterization. Consider the following transformation property [21] of Fourier transforms: if $\hat{f}(\mathbf{p})$ is the Fourier transform of $f(\mathbf{x})$ and $A \in \mathbb{R}^{2\times 2}$ an invertible matrix, then

$$\mathcal{F}f(A\mathbf{x}) = \frac{1}{|A|}\hat{f}(A^{-T}\mathbf{p}).$$

Hence, the MSE in target space reformulates to

$$\mathrm{mse}(\sigma_{\mathrm{i}}) = ||\frac{1}{|J_k^{-1}|}(\hat{f}_{\mathrm{i,ideal}}(J_k^T\mathbf{p}) - \hat{f}_{\mathrm{i,EWA}}(J_k^T\mathbf{p}))||_2.$$

An optimization will yield the same $\hat{\sigma}_{\mathrm{i}}$ as found for the source space optimization (set $\mathbf{p}' = J_k^T\mathbf{p}$). Intuitively, the transformation J_k^T will transform the optimal source interpolation covariance to the optimal destination interpolation covariance.

3) Anti-aliasing: The complete EWA resampling operation (4) is location-dependent, i.e., the EWA filter is a locally adaptive filter. Convolving the location dependent interpolation filter with an anti-aliasing filter results in a new EWA filter with location-dependent covariance matrix. Thus, the choice of an optimal V_{a} depends on the interpolation variance in target space: $J_k V_{\mathrm{i}} J_k^T$. That is, there is no single V_{a} that optimizes the EWA splatting operation. For instance, if we set the sum of σ_{i} and σ_{a} to $\hat{\sigma}_{\mathrm{i}}$, we have good filtering performance in regions where there is no scaling, but in areas with strong minifications, aliasing artifacts will appear. Setting σ_{a} larger introduces unnecessary blurring in regions with magnification (see Fig. 5(b,c)). In summary, V_{a} is locally adaptive and requires an adaptive anti-aliasing strategy.

B. Adaptive anti-aliasing

In the following, we derive a general closed form expression for the ideal adaptive anti-aliasing covariance matrix. Instead of using an MSE-based evaluation as used for the interpolation kernel, we analyze the resampling operation in frequency domain to derive the anti-aliasing covariance matrix. Aliasing occurs when the 2D frequency response of the transformed interpolation kernel $\mathcal{F} f_{\mathrm{i,EWA}}(J_k^{-1}\mathbf{x})$ is larger then the 2D Nyquist frequency [20]. More specifically, aliasing occurs when the frequency content exceeds the region delimited by the 2D rectangular function illustrated in Fig. 4(c). Our adaptive anti-aliasing strategy detects if such aliasing occurs, and locally adapts the non-isotropic anti-aliasing covariance matrix $V_{\mathrm{a}} = \mathrm{diag}([\sigma_{\mathrm{a},x}, \sigma_{\mathrm{a},y}])$ to avoid aliasing. In geometric terms, the corresponding principal ellipse axes are scaled to fit into the Nyquist rectangle. The principal axes of the ellipse are the eigenvectors of the covariance matrix (this property is denoted as *principal axis theorem*, valid for real symmetric matrices, see e.g. [22, p. 285]).

1) Detecting aliasing: To quantify the presence of aliasing, we evaluate the frequency response at the intersection of the principal axes of the transformed ellipse with the ideal anti-aliasing filter. If this frequency response value is large compared to the optimal Gaussian ($\hat{\sigma}_{\mathrm{i}}$), we will have aliasing. The transformation matrix $\tilde{C} = J_k V_{\mathrm{i}} J_k^T$ does reveal the transformation axes but not the principal axes of the target space Gaussian kernel (see Fig. 4(b,c)). The principal axes are obtained with an eigen decomposition: $\tilde{C} = Q\Lambda Q^T$, where Λ contains the magnitudes and Q the orthogonal directions of the principal axes.

We are interested in the intersection point of axis and ideal low-pass filter, hence, we only need the axes directions. One direction is given by $\alpha = q_{1,2}/q_{1,1}$, the second is $-\alpha^{-1}$ since the axes are orthogonal. Evaluating the decomposition yields

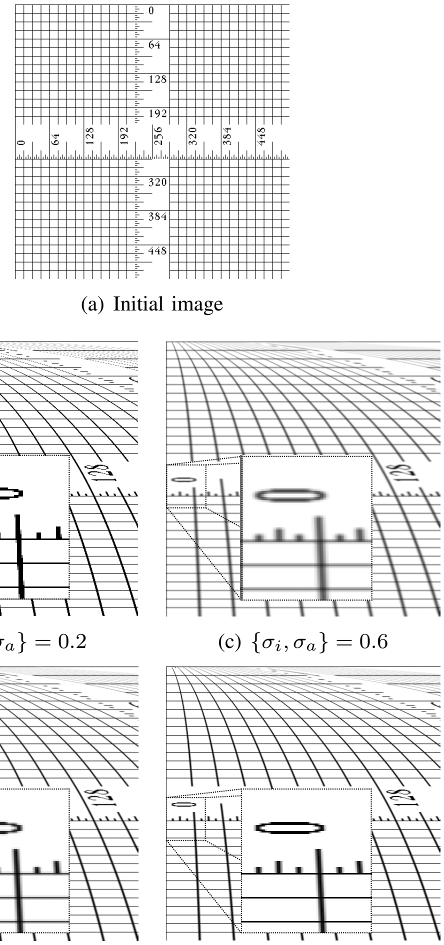
$$\alpha = \frac{2\tilde{c}_{1,2}}{\tilde{c}_{1,1} - \tilde{c}_{2,2} - \sqrt{4\tilde{c}_{1,2}^2 + (\tilde{c}_{1,1} - \tilde{c}_{2,2})^2}}.$$

Thus, the two intersection points of the EWA ellipse and the ideal low-pass filter are: $\mathbf{p}_1 = (1, \alpha)^T$ and $\mathbf{p}_2 = (-\alpha, 1)^T$ for $|\alpha| < 1$ or else $\mathbf{p}_1 = (\alpha^{-1}, 1)^T$ and $\mathbf{p}_2 = (1, -\alpha^{-1})^T$. If the value of the Gaussian filter at the intersection with an ideal low-pass filter is larger than the value of the optimal Gaussian kernel, there is aliasing. Hence, the condition for aliasing is

$$\exp(-1/2\mathbf{p}_l^T \tilde{C} \mathbf{p}_l) > \exp(-1/2\hat{\sigma}_{\mathrm{i}}^2) \quad l = 1, 2. \tag{6}$$

2) Removing aliasing: If the aliasing condition (6) holds, the interpolation kernel needs to be convolved with an anti-aliasing kernel. As stated earlier, this convolution leads to an addition of the covariance matrices $C = \tilde{C} + V_{\mathrm{a}}$. The anti-aliasing variance matrix can therefore be determined by substituting $\tilde{C}$ with C and by solving for the upper bound of the inequality (6)

$$\begin{aligned} \exp(-1/2\mathbf{p}_l^T(\tilde{C} + V_{\mathrm{a}})\mathbf{p}_l) &= \exp(-1/2\hat{\sigma}_{\mathrm{i}}^2), \quad l = 1, 2\ , \\ \mathbf{p}_l^T V_{\mathrm{a}} \mathbf{p}_l &= \sigma_{\mathrm{i}}^2 - \mathbf{p}_l^T \tilde{C} \mathbf{p}_l \quad l = 1, 2\ . \end{aligned}$$



(a) Initial image

(b) $\{\sigma_i, \sigma_a\} = 0.2$

(c) $\{\sigma_i, \sigma_a\} = 0.6$

(d) $\{\sigma_i, \sigma_a\} = 0.39$

(e) $\sigma_i = 0.39$, σ_a adaptive

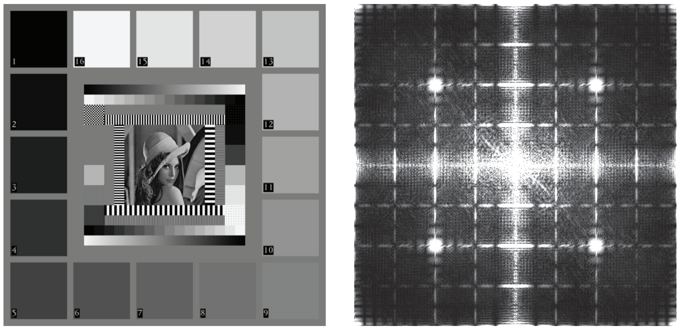
Fig. 5: *Non-linear, locally affine transformation of a test image, evaluated for different EWA filter parameterizations. Small uniform EWA variances reconstruct sharp images, but can lead to aliasing in areas of minification (b). Bigger EWA variances avoid these aliasing artifacts, but lead to excessive blurring in areas of magnification (c). Using the optimal uniform EWA variance leads to best tradeoff (d): aliasing is suppressed effectively, however, still some blurring is visible on the magnified areas. Our adaptive EWA formulation yields much sharper results in these areas (e), while still preserving the anti-aliasing filter properties in areas of minification.*

Combining the equation above with the condition for anti-aliasing yields

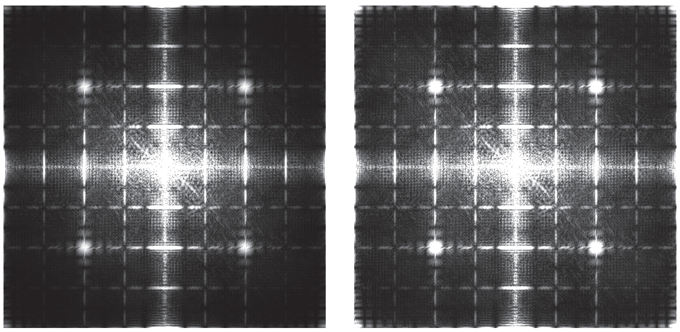
$$\begin{pmatrix} p_{1,1}^2 & p_{1,2}^2 \\ p_{2,1}^2 & p_{2,2}^2 \end{pmatrix} \begin{pmatrix} \sigma_{\mathrm{a},x}^2 \\ \sigma_{\mathrm{a},y}^2 \end{pmatrix} = \begin{pmatrix} \max(0, \hat{\sigma}_{\mathrm{i}}^2 - \mathbf{p}_1^T \tilde{C} \mathbf{p}_1) \\ \max(0, \hat{\sigma}_{\mathrm{i}}^2 - \mathbf{p}_2^T \tilde{C} \mathbf{p}_2) \end{pmatrix}, \tag{7}$$

where $\mathbf{p}_l = (p_{1,l}, p_{2,l})^T$, and $V_{\mathrm{a}} = \mathrm{diag}(\sigma_{\mathrm{a,x}}^2, \sigma_{\mathrm{a,y}}^2)$ represents the anti-aliasing covariance matrix. Thus, solving the expression for $\sigma_{\mathrm{a},x}^2$ and $\sigma_{\mathrm{a},y}^2$ provides the optimal choice for the EWA anti-aliasing filter in target space.

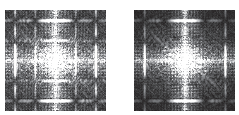
We evaluate the quality improvement of our adaptive anti-aliasing method using two different strategies: first, we provide



(a) Initial image (512x512) (spatial (left) and frequency (right) representation)



(b) One-to-one mapping (512x512). Left: $\{\sigma_i, \sigma_a\} = 0.39$, right $\sigma_i = 0.39$, σ_a adaptive. In the left image a considerable amount of blurring is visible (attenuation of high frequencies), which is quantified by a mean pixel error of approximately 3.3 between initial and rendered image; our adaptive strategy induces no blurring, initial and EWA rendered image are identical (mean error equals 0).



(c) Minification/Downsampling of 3.2 (160x160). Left $\{\sigma_i, \sigma_a\} = 0.2$, right $\sigma_i = 0.39$, σ_a adaptive. In the left image, aliasing is visible, our adaptive strategy successfully removes high-frequency components.

Fig. 6: *Frequency domain comparison of minification and one-to-one mappings of EWA rendering between fixed kernel sizes and our adaptive anti-aliasing strategy. The lowest frequency component is the center point of the magnitude plots.*

visual comparisons (Fig. 5) of different EWA parameterizations for a non-linear, locally affine transformation. As can be seen, using the ideal but constant parameterization for anti-aliasing and interpolation filters individually leads to blurring in magnified regions. Our adaptive EWA parameterization yields much sharper results in these areas, while still preserving the antialiasing filter properties in areas of minification. A second evaluation consists in comparing frequency responses after a one-to-one mapping and a minification with different EWA parameters. Fig. 6 shows that our adaptive strategy outperforms all other EWA parameters regarding over-blurring, illustrated with a one-to-one mapping, and aliasing, appearing for minifications.

3) Complexity reduction: For many video rendering applications, solving (7) for arbitrarily transformed covariance matrices $\tilde{C}$ is not necessary: often, the image transformation J_k only contains non-uniform scaling and no or very little shearing and rotations. More specifically, this holds true when the off-

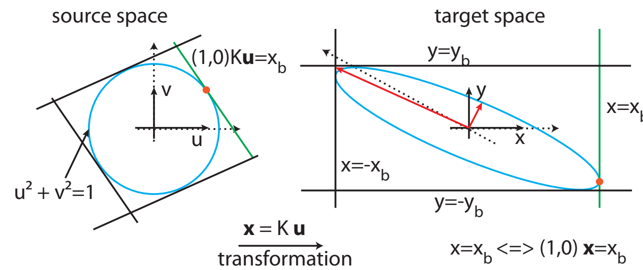


Fig. 7: *Geometric determination of the bounding box in target space. The intersection points of a specific ellipse level (e.g., exp(-0.5)) with a rectangular bounding box in target space are to be determined. Therefore the intersections are transformed to source space where the ellipse simplifies to a circle which makes the evaluation of the intersection easier.*

diagonal elements of $\tilde{C}$ are negligible compared to its diagonal elements. An evaluation of the locally affine transformations of several video retargeting examples shows that the off-diagonal elements are indeed several orders of magnitude smaller than the diagonal entries ($\tilde{c}_{i,j}/\tilde{c}_{i,i} \approx 10^{-3}$) on average and one order of magnitude in the worst case. Hence, $\tilde{C}$ behaves like a diagonal matrix, and the main directions of the ellipse are the principal axes $\mathbf{p}_1 = (1,0)^T$ and $\mathbf{p}_2 = (0,1)^T$. The anti-aliasing condition (7) can then be reduced to

$$\begin{aligned} \sigma_{\mathrm{a},x}^2 &= \max(0, \hat{\sigma}_{\mathrm{i}}^2 - \tilde{c}_{1,1}) \quad | + \tilde{c}_{1,1} \\ \underbrace{\tilde{c}_{1,1} + \sigma_{\mathrm{a},x}^2}_{c_{1,1}} &= \max(\tilde{c}_{1,1}, \hat{\sigma}_{\mathrm{i}}^2), \end{aligned}$$

and similar in y direction.

C. Bounding box

In theory, the contributions of the Gaussian filter need to be calculated over the entire image domain. In practice, however, the Gaussian weights decay very fast, and all weights falling below a pre-defined cut-off threshold can be discarded without noticeable image artifacts [13]. In the following, we will derive a tight axis-aligned bounding box that encloses the iso-line of a threshold value. All subsequent evaluations of the Gaussian will be limited to this bounding box.

Assume that we strive to limit the evaluation to a cut-off weight proportional to $\exp(-0.5)$. The EWA splatting equation (4) defines the implicit evaluation as $-0.5\mathbf{x}^T(J_k V_{\mathrm{i}} J_k^T + V_{\mathrm{a}})^{-1}\mathbf{x}$, where we omit the translational component without loss of generality. Unfortunately, this quadratic form does not directly reveal the explicit point transformation $\mathbf{x} = K\mathbf{u}$ which can be used to determine the exact bounding box. We therefore decompose $C = J_k V_{\mathrm{i}} J_k^T + V_{\mathrm{a}} = KK^T$ in order to obtain the transformation K. Since V_{i} and V_{a} are diagonal matrices, C is symmetric and can be diagonalized into an orthonormal basis [22]

$$C = Q\Lambda Q^T = KK^T,$$

where Q is orthogonal and Λ diagonal. Hence, $K = Q\sqrt{\Lambda}$ is uniquely obtained with the eigen decomposition.

Having obtained the explicit point transformation K, we can now derive the bounding box delimiters (see Fig. 7). The bounding box in target space is delimited by four straight lines $x = \pm x_{\mathrm{b}}$ and $y = \pm y_{\mathrm{b}}$, where $\mathbf{x} = (x, y)^T$ is a point in target space. Consider the case $x = x_{\mathrm{b}}$: the equation can be rewritten as $(1, 0)\mathbf{x} = x_{\mathrm{b}}$. Transforming the target coordinates back into source space yields

$$\begin{aligned} (1,0)K\mathbf{u} &= x_{\mathrm{b}} \\ k_{1,1}u + k_{1,2}v &= x_{\mathrm{b}}, \end{aligned} \tag{8}$$

with $\mathbf{u} = (u, v)^T$. Note that this expression resembles to a line equation in source space with normal vector $(1, -k_{1,1}/k_{1,2})^T$.

In source space, the EWA filter kernel resembles the unit circle. Hence, the optimal bounding box line must be tangent to the unit circle. This holds true because affine transformations conserve lines and intersections [23]. Expressed analytically,

$$u^2 + v^2 = 1, \tag{9}$$

$$(u, v)(1, -k_{1,1}/k_{1,2})^T = 0, \tag{10}$$

where the second equation is the condition for tangency. By combining (8), (9), and (10), we obtain

$$x_{\mathrm{b}} = \pm\sqrt{{k_{1,1}}^2 + {k_{1,2}}^2}.$$

Moreover, with

$$C = KK^T = \begin{pmatrix} k_{1,1}^2 + k_{1,2}^2 & \cdot \\ \cdot & k_{2,1}^2 + k_{2,2}^2 \end{pmatrix},$$

the bounding box equations simplify to

$$\begin{aligned} x_{\mathrm{b}} &= \pm\sqrt{c_{1,1}}, \\ y_{\mathrm{b}} &= \pm\sqrt{c_{2,2}}, \end{aligned}$$

where the second equation follows from a similar reasoning for y_{b}. The bounding box rectangle then delimits the ellipse to a cut-off value of $\exp(-0.5)$, since the rectangle delimits the unit circle in source space. For other cut-off values, the bounding box values can simply be scaled by s_{bb} such that $s_{\mathrm{bb}}x_{\mathrm{b}}$ and $s_{\mathrm{bb}}y_{\mathrm{b}}$ generate the desired cut-off values. Note that the same result can be derived by applying (parts of) the results from [24] to the 2D case.

IV. System Overview

This section puts the EWA splatting algorithm described above into a hardware context. To this end, we summarize first the specifications for the reference implementation. Subsequently, we tailor the data flow to the needs of a real-time streaming application and provide a top level view on the hardware architecture.

A. Specifications and Target Application

The EWA splatting setup is evaluated in the context of warp-based video retargeting [2]. This application involves large vertical and horizontal pixel deviations, which renders it an excellent test-case for developing an architecture that is able to handle even the most demanding warp kernels. Other applications, such as disparity mapping or multi-view generation, act much more locally and can therefore be considered a more simple special case.

Our implementation targets high-definition (HD) TV. The current HD TV standard is half HD (1280×720) at 25 frames per second (fps), denoted by 720p25. The implemented ASIC is designed to support 720p25, but its architecture is easily scalable to the upcoming full HD standard (1920×1080).

$k = 1 \dots W_s H_s$ source image index
$h = 1 \dots W_t H_t$ target image index
Input: pixel intensity w_k, Jacobian J_k, target position $m(\mathbf{u}_k)$, default variances $V_i = \sigma_i^2 I_2$, $V_a = \sigma_a^2 I_2$,
Output: $f_{EWA}(\mathbf{x}_h)$
for $k \in W_s H_s$ **do**
  Calculate $\tilde{C} = (J_k V_i J_k^T)$
  if *adaptive* **then**
    Determine V_a
  end
  Calculate $C^{-1} = (\tilde{C} + V_a)^{-1}$
  Calculate $c_k = 1/(2\pi)|J_k|\sqrt{|C^{-1}|}$
  Calculate bounding box : $\sqrt{c_{1,1}}, \sqrt{c_{2,2}}$
  for $\mathbf{x}_h$ *in bounding box* **do**
    $\phi = c_k \cdot \exp(-\frac{1}{2}(\mathbf{x}_h - m(\mathbf{u}_k)^T C^{-1}(\mathbf{x}_h - m(\mathbf{u}_k))$
    $\rho_h \leftarrow \rho_h + \phi$
    $\tilde{f}_{EWA}(\mathbf{x}_h) \leftarrow \tilde{f}_{EWA}(\mathbf{x}_h) + \phi \cdot w_k$
  end
end
for $h \in W_t H_t$ **do**
  Normalize $f_{EWA}(\mathbf{x}_h) = \tilde{f}_{EWA}(\mathbf{x}_h)/\rho_h$
end

Algorithm 1: Employed EWA splatting algorithm.

B. Data Flow

The original EWA rendering equation (4) describes the calculations to be performed for each output pixel. Unfortunately, a straightforward mapping of this equation is incompatible with the introduction of a bounding-box to reduce complexity, since identifying the subset of source pixels with relevant contributions to a pixel in the target image is an extremely complex task. Therefore, our approach reverses the flow such that we accumulate the contributions of each source pixel to the various different target pixels. The number of contributions of each source pixel can now easily be limited by the bounding box. However, due to this truncation and the fact that a Gaussian is not a real interpolation filter, a post-normalization step is required after the accumulation. Alg. 1 summarizes the main steps of the employed EWA splatting algorithm.

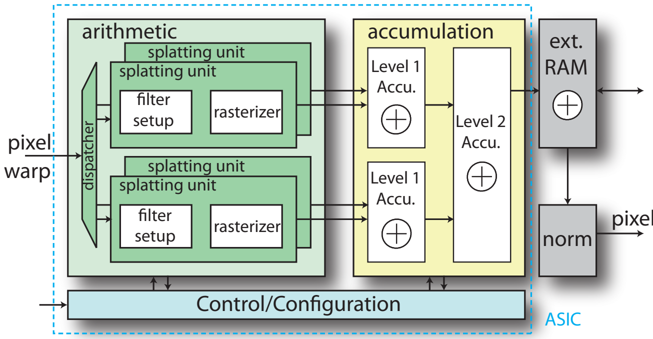


Fig. 8: *Architecture overview: the blue dotted line delimits the implemented part. In a fully working system, a specific external RAM interface needs to be added as well as the normalization unit.*

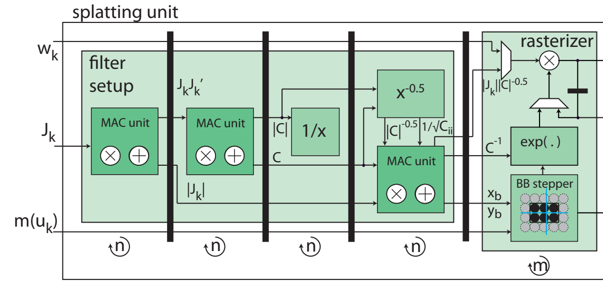


Fig. 9: *Architecture overview of EWA splatting unit (simplified). The black bars denote pipelines stages; each stage operates for n (filter setup) or m (rasterizer) cycles. The non-linear functions are detailed in Fig. 10.*

To match this modified data flow, we assume that pixels of the source video sequence are *streamed* row-wise into the architecture, for example through an HDMI or SDI interface. The output image is constructed and stored in a frame buffer, from where its pixels can be forwarded again to a standard video interface.

C. Architecture Overview

Fig. 8 provides a high-level view on the architecture. The inputs are the 8 bit gray-level[1] pixel (w_k) together with the the corresponding Jacobian transformation matrix (J_k) and the target pixel location ($m(\mathbf{u}_k)$). The output of the architecture are rendered pixels in the form of accumulated pixel contributions and corresponding accumulation weights for off-chip renormalization. We use a generic RAM interface and a handshake protocol to throttle the streaming input.

The architecture is subdivided into two main stages: an arithmetic part calculates the pixel contributions and an accumulation part collects and adds up the contributions in the target image. Since each source pixel contributes to several pixels in the target image, a caching architecture is proposed for the accumulation part to reduce the required memory bandwidth to the target frame buffer which serves as accumulator, thus requiring read-modify-write operations. Double buffering ensures that final read-out and accumulation do not collide.

V. EWA Splatting VLSI Architecture

In this section we provide details of the proposed architecture of the EWA splatting algorithm shown in Fig. 8 and in Alg. 1. We illustrate the involved hardware units and evaluate different number formats. We introduce a caching scheme that significantly reduces the required memory bandwidth.

[1]The architecture remains identical for 24 bit RGB color or 8 bit gray-level setups, only the total area and I/O bandwidth change to process three color channels in parallel.

A. EWA Arithmetic Part

The arithmetic part is structured into several parallel *splatting units* which allows for scaling the required splatting throughput easily. A dispatcher distributes the incoming Jacobian and pixel values to the different units. Each splatting unit first generates covariance matrix, determinants, and the bounding box in the *filter setup* stage and then *rasterizes* pixel contributions on an integer grid delimited by the bounding box (Fig. 9).

The filter setup stage is further divided into four pipelined and parallel running sub-stages, where each sub-stage is allocated a specific number of cycles before data is passed on to the next pipeline stage. Thus, the filter setup stage has a throughput determined by the number of cycles (eight cycles is found to be a good choice). The multi-cycle architecture is continued in the rasterizing part, where in each cycle one of the contributions within the bounding box is evaluated. The throughput can be increased or decreased by modifying the number of splatting units at constant AT-efficiency[2]. The benefit of the employed multi-cycle architecture over a fine-grained pipelined systolic architecture is the low pipelining area overhead and easy scalability with respect to the necessary throughput.

1) Datapath implementation: The datapath of a splatting unit contains linear matrix operations and several non-linear functions (Fig. 9 and Fig. 10). Two multiply-accumulate (MAC) units calculate the covariance matrix C and the required determinants $|J_k|$ and $|C|$. The reduced-complexity version of adaptive anti-aliasing comes at virtually no hardware overhead, since a thresholding operation merely consists of a comparator. The normalization factor $1/\sqrt{|C|}$ of the Gaussian is efficiently implemented with the fast inverse square root algorithm [25] using multiplications and additions. Similarly, the square roots in the bounding box calculation part $\sqrt{c_{i,i}}$ can be approximated with the fast inverse square root: $z \cdot 1/\sqrt{z}$. The bounding box is rounded to an integer grid such that the evaluation does not need to be very accurate. The 2-by-2 matrix inversion of the covariance matrix C is realized by multiplying the entries of C with $1/|C|$ (and inverting the sign of the off-diagonal elements). The inversion is realized with a coarse look-up table

[2]The area-delay (AT) product is a standard metric in digital VLSI to compare efficiency of hardware architectures.

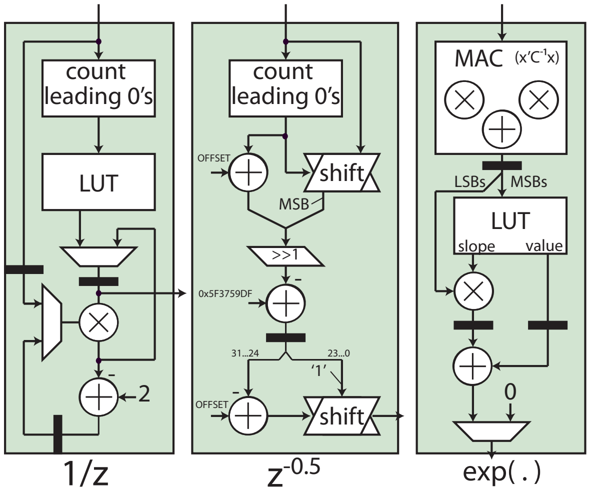


Fig. 10: *Architecture of non-linear function approximations. The constant 'OFFSET' of the fast inverse square root block depends on the integer width of the input ($= 127 - \text{width} + 1$), for an explanation of the value 0x5F3759DF see [25].*

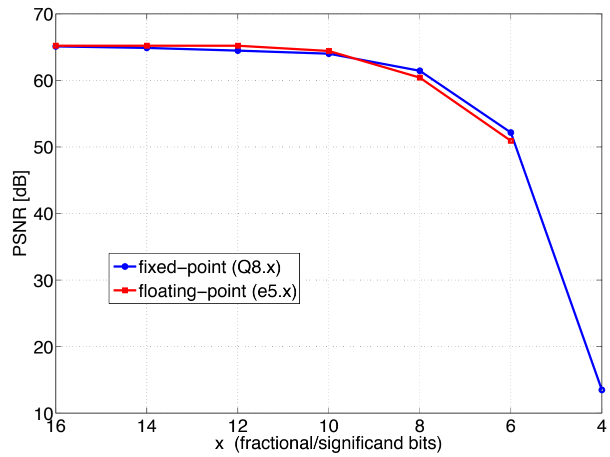


Fig. 11: *PSNR of different number formats illustrated with one typical example image. The abbreviation $Qa.b$ stands for fixed point format with a integer bits and b fractional bits, $ea.b$ stands for floating-point with an a bit exponent and b bits significand. The plot shows $Q8.x$ and $e5.x$, where x is determined by the x-axis of the plot.*

(LUT) followed by four refining Newton iterations

$$z_{n+1} = 2 \cdot z_n - z_n^2 \cdot a, \qquad n = 0, \ldots, 3$$

where a is the value to be inverted, z_n the result after n iterations, and z_0 is the initial LUT value. The sampling points of the values in the LUT are logarithmically spaced over the function domain to increase precision. The (base-2) logarithmic look-up is obtained by addressing the number of leading zeros. A third MAC unit realizes the multiplications of $c_{i,j}$ by $1/|C|$, the bounding box scaling $s_{\text{bb}}\sqrt{c_{i,i}}$, and the multiplication of the normalization factors $1/(2\pi)|J_k||C|^{-0.5}$. Finally, the exponential function in the rasterizer part is realized with linear interpolation between uniformly spaced pre-calculated supporting points.

2) Arithmetic precision: To quantify the precision of the non-linear function approximations described in the previous paragraph and to decide on the most efficient number format in view of chip area and throughput, we compare various number formats against the IEEE single precision standard (32 bit floating point). The simulated number formats are custom fixed point and custom floating point formats defined by the number of integer/fraction bits and significand/exponent bits, respectively.

The PSNR for the complete EWA splatting system of the custom number formats compared to single precision is shown in Fig. 11. The specified data format is used for all arithmetic operations in the data-path except for some well-defined signals such as I/Os. Due to the non-linear approximations, the PSNR ceils at a certain value. The ceil value, between 60 dB and 70 dB, is sufficiently high to conclude that the approximations do not have a noticeable impact on quality. Also, fixed-point and floating-point formats converge to the same PSNR value, such that both formats are equivalent in terms of precision for the corresponding number formats. In terms of AT-efficiency, the fixed-point variant performs slightly better when comparing a MAC unit with number formats at the same PSNR value. Thus, we opted for a fixed point architecture.

B. Accumulation and Caching

The main challenge of the accumulation part is the increased bandwidth requirement on the framebuffer memory compared to the bandwidth of the input pixels since the rasterizers generate several pixels for each input pixel. Moreover, the pixels to be accumulated over time are, in principle, arbitrary distributed in the target image such that we need to buffer and access the entire target image in random-access fashion. To reduce the bandwidth and random-access pattern to the frame buffer, the accumulation is realized in several stages which takes the form of a *two-level cache*.

1) Key observations: The rectangular bounding boxes access memory in blocks of neighboring pixels. Besides, since source pixels are streamed in scanline order, subsequent target pixels typically exhibit large horizontal overlaps. The same holds for vertically neighboring pixels. Our proposed accumulation architecture therefore first absorbs neighboring contributions both horizontally and vertically and then writes larger chunks of partially accumulated pixels into the external memory. Also, since accumulation is mathematically an associative and commutative operation, the complete accumulation operation can be split into partial accumulations. This property allows to separate the accumulation into multiple accumulation stages.

2) Analysis of accumulation bandwidth: The warp function usually transforms neighboring pixels from the input image into neighboring pixels in the output image, which leads to strong spatial correlation in the accesses to the off-chip frame buffer. Fig. 12 provides an example for such an access pattern. The strong correlation motivates an on-chip accumulation

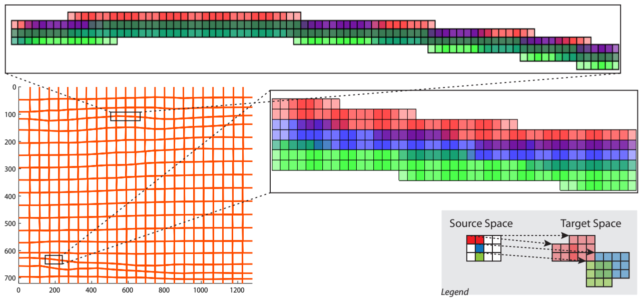


Fig. 12: *Typical accumulation buffer access patterns for the warped pixels positions. The horizontal scanline order of the source image is transformed to curved scanline orders in the target image, and no holes are formed by the continuous warp function. Therefore, consecutive horizontal and vertical target pixels will overlap. The zoomed windows show the incoming pixel bounding boxes in detail, for three consecutive lines (red, green, and blue). Overlaps are indicated by color blending of the different colors (see legend).*

buffer that stores several lines to take advantage of both horizontal and vertical proximity. The effect of on-chip buffers on external bandwidth is shown in Fig. 13. The average splat size being a window of almost 3-by-3 pixels, we see that with an on-chip buffer smaller than the average splat size, we need almost 9 times the bandwidth compared to writing the image once. If the on-chip buffer is larger than the average splat size, the horizontal proximity is exploited and thus the bandwidth is reduced by a factor of 3. Finally, if the on-chip buffer covers more than 3 vertical lines, vertical proximity can be exploited and the minimal possible bandwidth is approached.

In addition to the on-chip accumulation buffer, two additional design choices are made to increase the accumulation performance. The first design choice is motivated by the nature of the non-linear warping, where horizontal source lines can possibly be rendered to (almost) arbitrary-shaped curves. If the on-chip buffer is partitioned into multiple lines that span the full image width, then variations in the vertical direction beyond the number of lines within the on-chip buffer requires costly swap operations. We therefore split the accumulation buffer into many two-dimensional blocks (tiles), where each block can be individually addressed. The impact of block size and total buffer size on bandwidth performance is summarized in Fig. 14: the smaller the block size the lower the required accumulation bandwidth but at the cost of higher address complexity and hence increased critical path and chip area. The specific block size configuration of our architecture will be detailed in the next section.

The second design choice is motivated by the size of the accumulation buffer, which is typically on the order of kBytes to MBytes, and therefore needs to be realized using bandwidth-limited on-chip SRAM blocks. While using such on-chip buffers reduces the external memory bandwidth, it also shifts the bandwidth bottleneck from external memory to the on-chip memory blocks. We therefore introduce another level of smaller-sized accumulation buffers, which can be realized

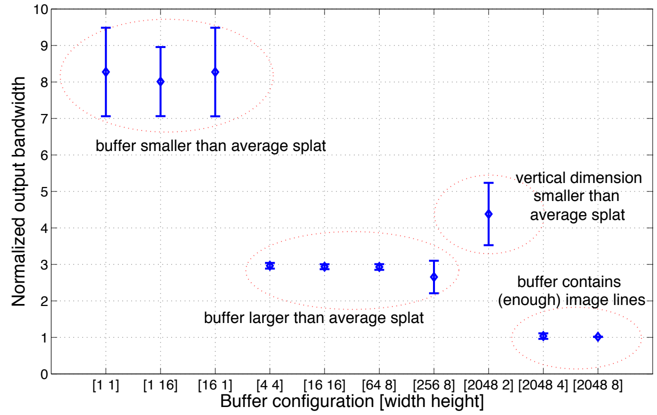


Fig. 13: *Bandwidth simulation of different sizes of the on-chip accumulation buffers, performed for different scenes. The plot shows the mean values with associated vertical error bars to indicate the standard deviation. The accumulation buffer model is assumed to be ideal, where each tile is comprised of 1x1 pixel (cf. Fig. 14). The external output bandwidth is normalized to the bandwidth required to transmit one full frame buffer once, and therefore a bandwidth of 1 can be considered optimal.*

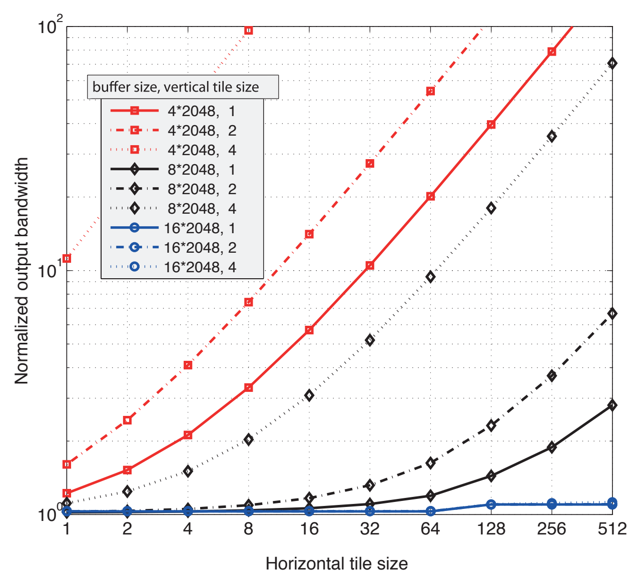


Fig. 14: *Effect of tile size and total on-chip buffer size on external memory bandwidth, simulated for 720p images. The external output bandwidth is normalized to the bandwidth required to transmit one full frame buffer once, and therefore a bandwidth of 1 can be considered optimal. For our design, we select a horizontal tile size of 32 and a vertical tile size of 2, with an overall on-chip buffer size of 8*2048 pixels for 720p or 1080p images.*

in high-bandwidth distributed memory resources that allow concurrent access to all elements. Note that this two-level approach has no impact on external bandwidth.

3) Architecture details: As motivated before, our architecture employs a multi-level accumulation strategy. In a first stage

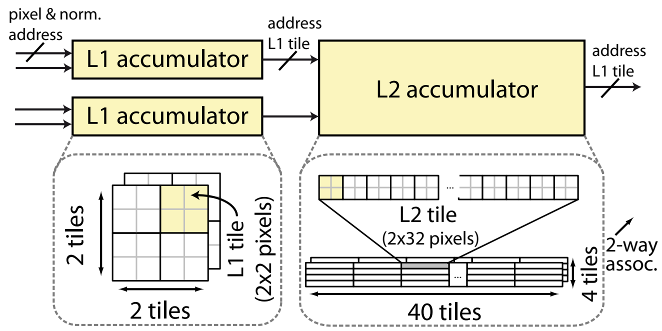


Fig. 15: *Two-level accumulation architecture, and resulting cache configuration for 720p. A third accumulation level (not shown in the figure) is realized in the frame buffer.*

neighboring and overlapping pixels of the rasterizers are combined into several small and register-based accumulation buffers. Next, the resulting chunks of spatially adjacent pixels are accumulated in larger SRAM-based accumulation buffers which can exploit larger vertical pixel proximity by absorbing several target lines. In a final step, the outputs are transferred and accumulated in the frame buffer (external memory). In the best case, the last step does not require an accumulation but only a write-out operation. The proposed structure is shown in Fig. 15.

The two on-chip accumulators are detailed in the following using cache terminology, i.e., they are referred to as level 1 (L1) and level 2 (L2) caches, respectively. Each cache is composed of multiple tiles (blocks of pixels). The particularity of our caches is that we do not replace the cache tiles with data but rather empty them and accumulate the content in the next higher hierarchy. The address conversion from external memory to the accumulation buffer is performed using a *2-way set associative* mapping. Table I summarizes potential cache configurations for various resolutions.

Using 2-way set associative accumulation buffers, each pixel address can potentially be mapped to two different cache tiles. In comparison, direct address mapping uses pre-determined addresses for each external pixel address. In theory, set associativity reduces the number of address collisions and increases the flexibility of a cache, at the cost of an overhead in storage of address tags and of increased addressing complexity. In our case, the overhead of using 2-way set associativity is negligible, but it also has only a minor effect on bandwidth. A bigger advantage is the possibility to efficiently balance the L1-L2 transfers: if one of the two blocks within a set contains a partial accumulation result, it gets flagged for swapping. If several blocks contain one (or two) partial accumulation result, an additional least recently used flag determines the swap priority. This allows to continuously transfer data from the L1 to the L2 accumulation stage and thus minimizes cache write-out misses. This way, cache transfers can be balanced better to achieve constant bandwidth at full capacity. In summary, the 2-way associativity provides an efficient mechanism to determine blocks that are most likely to produce a cache write-out miss.

TABLE I: *Cache configurations and sizes for different target resolutions. All the resolutions have widescreen 16:9 aspect ratio, the number indicating the vertical resolution. 'g' stands for gray level images, 'c' for color images. The last row (4 lines, g) indicates the cache used in the implemented ASIC. The cache is 2-way set associative.*

	L1 cache	L2 cache		
memory type	flip-flop array	dual-port SRAM macro		
read ports (g/c)	32/64 & 128/256 bit	128/256 bit		
write ports (g/c)	32/64 bit	128/256 bit		
tile/block size	2×2 pixels	2×32 pixels		
		576p	**720p**	**1080p**
# tiles (8 lines)	8	256	320	480
size (8 lines, c)	256 bytes	128kB	160kB	240kB
size (4 lines, g)	*128 bytes*	*32kB*		

VI. Results

In this section, we summarize the results, both in terms of rendering quality and implementation results, and also recapitulate the limitations of our architecture.

A. Throughput

For the following throughput evaluation we use a clock frequency of 133 MHz (synthesis result) and a frame rate of 25 (e.g., 720p25). One input pixel is assumed to generate 9 output pixels and each pixel is a 24 bit RGB value.

1) 720p: One splatting unit has a throughput of $6.65\ \mathrm{MPixels/s}$. An 720p25 video stream requires a throughput of $23\ \mathrm{MPixels/s}$ and thus four splatting units. The necessary external memory bandwidth without caching is $2 \times 9 \times 23\mathrm{MPixels/s}$ which amounts to 3.31 GB/s for 8 byte per pixel (accumulated RGB values plus normalization weight). The factor 2 comes from the read-modify-write operation of the accumulation. A cache efficiency of 83% (2×1.5 the minimal bandwidth) reduces the bandwidth to 550 MB/s. An additional read/clear operation to the memory is further necessary to account for the final read-out and clearing. An L2 cache of 8 lines with 1280 pixels per line (160 KB) is necessary to reach the targeted cache efficiency (see Fig. 14).

2) 1080p and beyond: For 1080p, the number of splatting units and L1 caches needs to be doubled and the L2 cache needs to be extended to line size of 1920 (240 KB). Also, such an architecture allows for rendering 720p at higher frame rates (720p50, 720p60). The architecture can be extended to resolutions beyond 1080p if the interface bandwidth between L1 and L2 cache is scaled accordingly.

3) Comparison to software implementation: In order to put these numbers into context, we provide performance tests results of EWA rendering on a high-end CPU. The computation time depends on the chosen image resolution and the warp type. For 720p, our C++ based implementation takes between 155 ms and 165 ms for different video retargeting sequences on a high-end machine equipped with an Intel XEON 3.2 GHz (W3565) processor and 24 GB RAM.

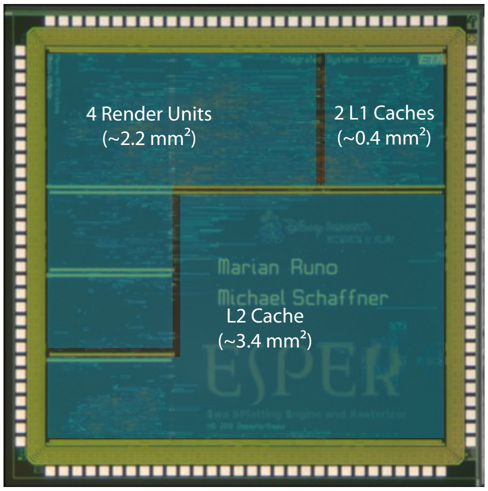


Fig. 16: *EWA video rendering chip micrograph with overlayed main blocks and their corresponding size.*

B. CMOS implementation results

The architecture described in Sec. V was implemented in VHDL and was fabricated in 180nm (1P6M) CMOS technology. A chip micrograph is provided in Fig. 16. The design supports image resolutions up to 2048×2048. It employs four splatting units to support 720p25 in splatting performance. The implemented L2 cache is reported in Tbl. I: due to die size limitations the cache is reduced to 4 lines of gray-valued 576p.

The ASIC has been successfully tested at 123 MHz where a power consumption of 300 mW has been measured. Core voltage is 1.8 V and I/O pad voltage is 3.3 V. Core area is 6 mm^2 which corresponds to 660 kGE. There are 64 data I/O pins and 56 power/ground pins.

C. Rendering results

To illustrate the purpose and quality of EWA rendering, we provide an example of 2D image retargeting. The images in Fig. 17 show an initial image with aspect ratio of 16:9, the content-aware retargeted 4:3 version, and the linear scaled version for reference. The warps have been generated with a framework similar to [2]. For more examples and explanations on video retargeting refer to [2] directly.

D. Discussion of temporal aspects

The target applications of the EWA rendering architecture are real-time video applications. To render video, i.e., a sequence of correlated images, it is often not sufficient to render the images individually, but temporal effects need to be taken into account. Temporal artifacts occur when objects within a video sequence are warped inconsistently in consecutive images. A prominent example is a non-moving object that is warped into different positions in consecutive frames which would be perceived as 'wobbling effect'. Fig. 18 shows an example of such a temporal artifact. Note that this artifact is not specific to EWA splatting but a general problem in video rendering. An efficient solution is to constraint the warp grid by minimizing a temporal coherence energy expression ([26], [2])

$$\sum_{k \in D_s} (m_t(u_k) - m_{t-1}(u_k))^2,$$

which penalizes varying warp positions over time. The formula uses the same notation as in Sec. II with the additional image index t. Fig. 18 compares several video frames rendered with and without this temporal stabilization. More details on temporal stability can be found in [2].

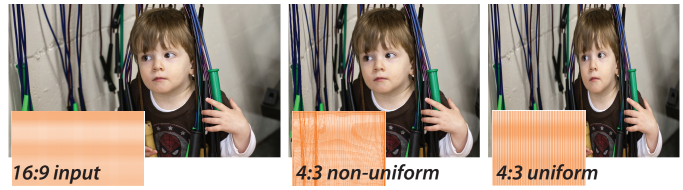


Fig. 17: *Example of aspect ratio retargeting [2]. The retargeted image and associated image warp are shown in comparison to uniform image scaling: while uniform scaling visually distorts the image, the retargeting algorithm keeps the visually important portions of the image undistorted. We show a conversion from 16:9 to 4:3 aspect ratio (Image courtesy of Andrew Malone).*

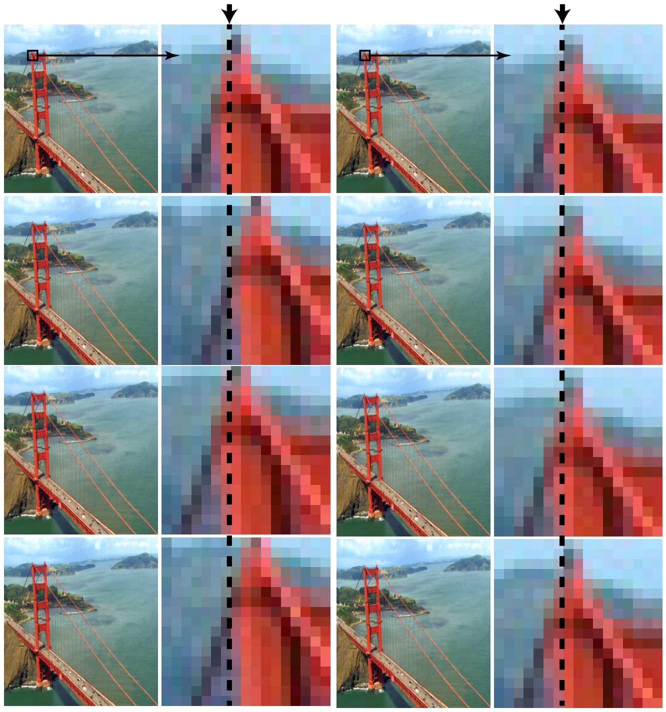
E. Limitations and future work

Algorithmically, the following limitations have to be taken into account and potentially addressed in future work. First, the linear approximation of the per pixel warp function is only able to handle locally affine transformations correctly. Besides, the EWA framework always introduces a tradeoff between aliasing and blurring, which might be improved with different warping approaches. Finally, our implemented simplified adaptive anti-aliasing strategy leads to aliasing when the warping consists of significant rotations and shearing.

The VLSI architecture has not been optimized for low-power operation so far. Although clock gating has been included, no design effort has been spent to make the design specifically low-power. An improved CMOS implementation will account for this. Also, we investigate techniques for lowering the required cache size.

VII. Conclusion

EWA splatting is a promising technique for current and next-generation HD video applications such as video retargeting, disparity mapping, and multi-view synthesis. Setting the Gaussian filter variances in an adaptive way greatly improves rendering quality. Thus, with the proposed adaptive strategy, we are able to render high-quality images without aliasing or excessive blurring. Furthermore, we show that EWA rendering can be efficiently implemented into a VLSI circuit, which would be targeted for end-user display integration. The proposed VLSI architecture for real-time EWA splatting provides



No temporal coherence   With temporal coherence

Fig. 18: *Example of temporal stability of aspect ratio retargeting. Four consecutive frames of a video sequence with slow camera are shown, both without temporal coherence constraints and with temporal coherence constraints. To illustrate the effect of the coherence constraints better, we show a zoomed portion of the image with a vertical line as column reference. Without temporal constraints, the top of the bridge slightly moves left and right, which is perceived as a 'wobbling effect'. Using temporal constraints, such erroneous motion can be suppressed effectively. (Image sequence: ©Mammoth HD).*

high-quality results using fixed-precision number formats. Multi-level accumulation significantly reduces the necessary memory bandwidth to the external frame buffer.

References

[1] H. Schwarz, D. Marpe, and T. Wiegand, "Overview of the scalable video coding extension of the H.264/AVC standard," *Circuits and Systems for Video Technology, IEEE Transactions on*, vol. 17, no. 9, pp. 1103–1120, 2007.

[2] P. Krähenbühl, M. Lang, A. Hornung, and M. Gross, "A system for retargeting of streaming video," *ACM Trans. Graph.*, vol. 28, no. 5, pp. 126:1–126:10, 2009.

[3] A. Smolic, P. Kauff, S. Knorr, A. Hornung, M. Kunter, M. Mueller, and M. Lang, "Three-dimensional video postproduction and processing," *Proceedings of the IEEE*, vol. 99, no. 4, pp. 607–625, 2011.

[4] M. Lang, A. Hornung, O. Wang, S. Poulakos, A. Smolic, and M. Gross, "Nonlinear disparity mapping for stereoscopic 3D," *ACM Trans. Graph.*, vol. 29, no. 4, pp. 75:1–75:10, 2010.

[5] C.-H. Chang, C.-K. Liang, and Y.-Y. Chuang, "Content-aware display adaptation and interactive editing for stereoscopic images," *Multimedia, IEEE Transactions on*, vol. 13, no. 4, pp. 589–601, 2011.

[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), 2011 IEEE International Conference on*, 2011.

[7] M. Tanimoto, M. Tehrani, T. Fujii, and T. Yendo, "Free-viewpoint TV," *Signal Processing Magazine, IEEE*, vol. 28, no. 1, pp. 67 –76, 2011.

[8] A. Smolic, "3D video and free viewpoint video: From capture to display," *Pattern Recognition*, vol. 44, no. 9, pp. 1958–1969, 2011.

[9] M. Do, Q. Nguyen, H. Nguyen, D. Kubacki, and S. Patel, "Immersive visual communication," *Signal Processing Magazine, IEEE*, vol. 28, no. 1, pp. 58 –66, 2011.

[10] T. Akenine-Moller, E. Haines, and N. Hoffman, *Real-time rendering*. AK Peters, 2008.

[11] T. Weyrich, S. Heinzle, T. Aila, D. Fasnacht, S. Oetiker, M. Botsch, C. Flaig, S. Mall, K. Rohrer, N. Felber *et al.*, "A hardware architecture for surface splatting," *ACM Trans. Graph.*, vol. 26, no. 3, pp. 23:1-23:9, 2007.

[12] S. Heinzle, G. Guennebaud, M. Botsch, and M. Gross, "A hardware processing unit for point sets," in *Proceedings of the 23rd ACM SIGGRAPH/EUROGRAPHICS symposium on Graphics hardware*. Eurographics Association, 2008, pp. 21–31.

[13] P. Heckbert, "Fundamentals of texture mapping and image warping," Masters Thesis, Univ. of California, Berkeley, Dept. of Electrical Eng. and Computer Science, 1989.

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

[15] R. Szeliski, S. Winder, and M. Uyttendaele, "High-quality multi-pass image resampling," Microsoft Research, Tech. Rep., 2010.

[16] B. Triggs, "Empirical filter estimation for subpixel interpolation and matching," in *Computer Vision, 2001. ICCV 2001. IEEE 8th International Conference on*, vol. 2, 2001, pp. 550–557.

[17] R. Stasinski and J. Konrad, "Improved POCS-based image reconstruction from irregularly-spaced samples," in *Proceedings of the XI European Signal Processing Conference*, 2002, pp. 271–290.

[18] Y.-R. Horng, Y.-C. Tseng, and T.-S. Chang, "VLSI architecture for real-time HD 1080p view synthesis engine," *Circuits and Systems for Video Technology, IEEE Transactions on*, vol. 21, no. 9, pp. 1329 –1340, 2011.

[19] M. Zwicker, H. Pfister, J. V. Baar, and M. Gross, "EWA splatting," *Visualization and Computer Graphics, IEEE Transactions on*, vol. 8, no. 3, pp. 223–238, 2002.

[20] R. Gonzales and R. Woods, *Digital Image Processing*. Prentice Hall, 2002.

[21] R. Bracewell, K.-Y. Chang, A. Jha, and Y.-H. Wang, "Affine theorem for two-dimensional fourier transform," *Electronics Letters*, vol. 29, no. 3, p. 304, 1993.

[22] G. Strang, *Linear Algebra and Its Applications*. Brooks Cole, 2005.

[23] R. Hartley and A. Zisserman, *Multiple view geometry*. Cambridge university press Cambridge, UK, 2000.

[24] C. Sigg, T. Weyrich, M. Botsch, and M. Gross, "GPU-based ray-casting of quadratic surfaces," in *Eurographics Symposium on Point-Based Graphics*, 2006, pp. 59–65.

[25] C. Lomont, "Fast inverse square root," Purdue University, Tech. Rep., 2003. [Online]. Available: http://www.lomont.org/Math/Papers/2003/InvSqrt. pdf

[26] L. Wolf, M. Guttmann, and D. Cohen-Or, "Non-homogeneous content-driven video-retargeting," in *Computer Vision, 2007. ICCV 2007. IEEE 11th International Conference on*, 2007.