REAL-TIME HARDWARE IMPLEMENTATION OF MULTI-RESOLUTION IMAGE BLENDING

Vladan Popovic, Kerem Seyid, Alexandre Schmid, Yusuf Leblebici

Ecole Polytechnique Fédérale de Lausanne (EPFL)
Microelectronic Systems Laboratory
Lausanne, Switzerland

ABSTRACT

A novel real-time implementation of a multi-resolution image blending algorithm is presented in this paper. A multi-resolution decomposition of the input is used to blend multiple images at different scales. Processing time is shortened by designing a pipeline system. The proposed solution requires less hardware multipliers and is able to achieve very high operating frequencies, compared to the current designs. The presented hardware architecture is optimized to support multiple simultaneous video streams, and high frame rates at High-Definition (HD) resolutions.

Index Terms— Real-time systems, Pipeline processing, Image fusion, Image decomposition, Field programmable gate arrays

1. INTRODUCTION

The limited angle of view of modern cameras has created the need to combine two or more images into a single one, in order to increase the effective angle of view. The creation of panoramas or image mosaics has been a popular research topic over the past years. The problems which must be solved relate to the proper image alignment and seamless image blending.

The purpose of the image alignment is to determine the correct orientation and position of the original images in the final mosaic. Various algorithms for aligning the captured images were developed [1], [2], [3]. Additionally, it is possible to reconstruct a panoramic mosaic using a video stream of frames [4]. While image alignment processes the geometry of the image, blending algorithms handle the pixel intensity in the final mosaic.

A major issue in creating photo-mosaics resides in the fact that the original images do not have identical brightness levels. This may be caused by diverging camera orientations in space. Thus, cameras acquire more light in some of the shots. The problem manifests itself by the appearance of a visible seam in regions where the images overlap. The blending algorithms based on a weighted average between pixels in every image, *e.g.* "Cut and paste" algorithm [4], can reduce or even completely remove the seams. However, the drawback of a weighted average lies in a high frequency blurring in the presence of any small image alignment error. A possible solution to this issue consists of using a multi-resolution blending algorithm [5], [6] where high frequencies are combined in a small spatial range, thus avoiding blurring.

Blending is usually realized as a post-processing operation on a Personal Computer (PC). However, real-time blending is often required in multi-camera systems, *e.g.* [7], [8]. Real-time operation can be a very challenging problem. Hence, a Graphics Processing Unit (GPU) implementation or a dedicated hardware solution are often considered. Various existing GPU implementations of multi-resolution blending algorithms [9], [10] and their performance will be compared to this work. On the other hand, Field Programmable Gate Arrays (FPGA) are widespread used platforms, that enable fast development. Sims and Irvine [11] designed an FPGA system for blending using gradient pyramids. However, their system targeted blending greyscale images with VGA resolution. Furthermore, the system had large memory requirements, because all of the temporary results in the calculation process had to be stored. Song et al. [12] introduced a resource-efficient three-stage pipeline processing system. Still, their system only supports dual channel image fusion and is also constrained to greyscale images with VGA resolution. Finally, Van Der Wal and Burt [13] designed an Application Specific Integrated Circuit (ASIC) able to decompose an input image into multiple resolutions. However, multiple processing and memory chips have to be used in order to blend images.

In this paper, a novel real-time FPGA-based implementation is presented. The dedicated hardware for the multi-resolution blending algorithm is implemented using a fully pipelined architecture. The requirements for storage elements is reduced since only the final results are stored into memory. Furthermore, the design is able to support higher frame rate and higher image resolution than earlier proposed systems.

The outline of the paper is as follows. An overview of the multi-resolution image blending is given in Section 2. The formulation of the problem and proposal of the new implementation are explained in Section 3. Finally, the experimental results and comparison to the related work are presented in Section 4.

2. MULTI-BAND BLENDING

Multi-Band Blending (MBB) [3] is based on a multi-resolution decomposition of the original images and their blending across octave frequency bands. The images are represented using a Laplacian Pyramid (LP) [5], as it has perfect and simple reconstruction [14]. Several steps are performed to obtain the desired LP. The image is first blurred and then downsampled by a factor of 2 to obtain a low-pass image. The low-pass filter proposed in [14]:

$$H(z) = G(z) = \frac{1}{16}(1 + 4z^{-1} + 6z^{-2} + 4z^{-3} + z^{-4}) \quad (1)$$

has a very high precision, since it uses only integer coefficients. Furthermore, this filter can be implemented in hardware using only adders and shifters, which is further detailed in Section 3.

The low-pass image is then upsampled by 2 and reconstructed using an interpolation filter. The interpolation filter, in this case, is

The authors gratefully acknowledge the support of XILINX, Inc., through the XILINX University Program.

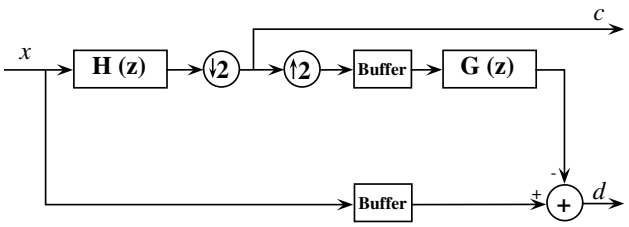

Fig. 1. Two-level LP decomposition

identical to (1). The interpolated image is subtracted from the original to determine a high-frequency version of the input image. The LP is created by repeating the same procedure on the downsampled low-frequency image.

The regions of overlap between images may be of an arbitrary shape. Thus, a mask should be created, defining the pixels which should be taken from the original image and their respective weight. A binary mask is assigned to each image, where 1 represents a pixel that should be taken from the selected image. This mask is further decomposed into a Gaussian Pyramid (GP), which is created by repetitive blur and downsample operations, *i.e.* each level of the pyramid is a low-pass version of the previous level. Brown [5] and Burt [3] suggest to use the same filter for the generation of the GP weight mask as for the LP. The use of the same filter simplifies the system and provides seamless blending results when the overall brightness level of the images does not differ significantly.

Each frequency band of the LP is combined with the respective frequency band of the other LPs, *i.e.* other images. A weighted average is applied within the overlapped areas, which are proportional in size to the wavelengths represented in the band. Hence, when coarse features occur in the overlapping region, they are gradually blended over a relatively large distance, without blurring or degrading finer image details in the neighborhood [5]. The weights are taken from the corresponding mask GP. In case of blending two images, A and B, the blending of one pyramid level is expressed as:

$$I(x,y) = I_A(x,y)w(x,y) + I_B(x,y)(1 - w(x,y)) \qquad (2)$$

where I_A and I_B are pixel intensities and w is the pixel weight.

3. FPGA IMPLEMENTATION

3.1. Laplacian Pyramid Decomposition

In this paper, we propose an FPGA design of the multi-resolution image blending based on LP decomposition. A fully pipelined architecture is utilized. Figure 1 shows the data flow diagram of two-level LP decomposition. The obtained results are coarse (c) and detail (d) images. Filters from (1) can be expressed in the spatial domain by the matrix:

$$H = G = \frac{1}{256}\begin{bmatrix} 1 & 4 & 6 & 4 & 1 \\ 4 & 16 & 24 & 16 & 4 \\ 6 & 24 & 36 & 24 & 6 \\ 4 & 16 & 24 & 16 & 4 \\ 1 & 4 & 6 & 4 & 1 \end{bmatrix} \qquad (3)$$

Direct two-dimensional filtering is implemented, since it requires fewer buffers for storage of temporary results than separable filtering. Even though a 5×5 pixel window is needed for filtering with $G(z)$, at least two rows (columns) are filled with zeros after the upsampling operation. Hence, the buffer following the upsampler in

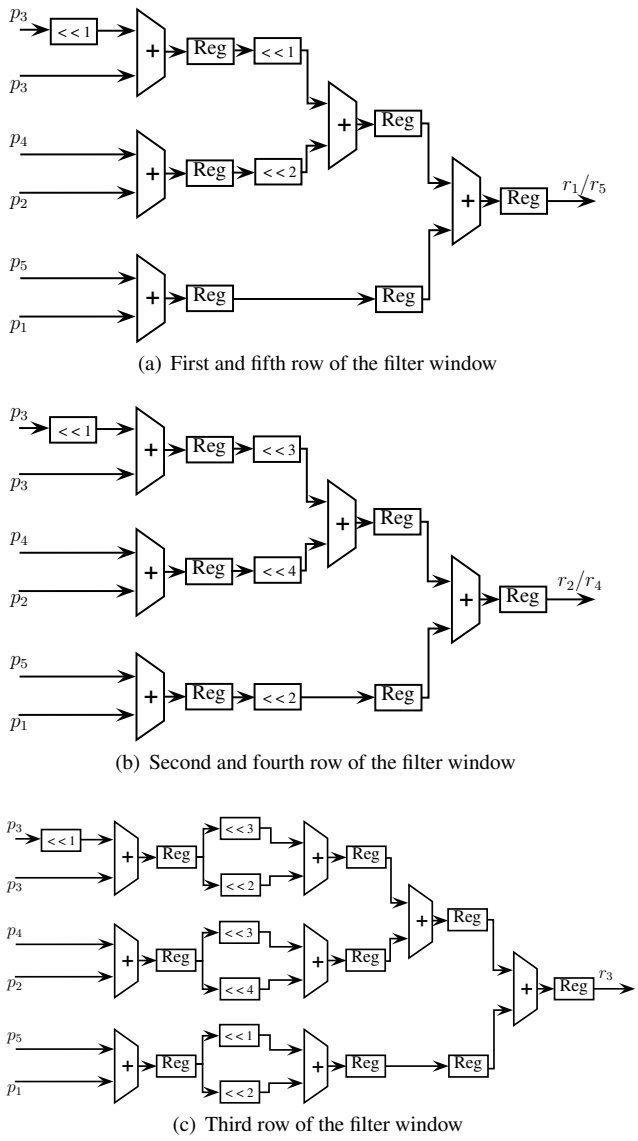

Fig. 2. Low-pass filter implementation

Figure 1 only stores two rows (columns). The interpolation filtering starts when the third non-zero row (column) pixels are arriving from the pipeline. The buffer in the lower branch in Figure 1 has the same depth as the buffer located in the upper branch, and acts as a delay element which synchronizes the original image with the interpolated pixels. The full decomposition into an LP is realized by cascading the proposed implementation. The number of the cascaded blocks corresponds to the desired number of LP levels of decomposition.

The acquired images are temporarily stored into a Random Access Memory (RAM). The LP decomposition and filter implementation depend on the order of the pixels which are read from the RAM. The first five pixels of each row are read consecutively, *i.e.* five pixels are read from the first row, followed by five pixels from the second row, etc. Subsequently, the filter window is shifted by two columns to the right. By reading from the memory in this manner, both low-pass filters $H(z)$ and $G(z)$ in Figure 1 provide outputs column-by-

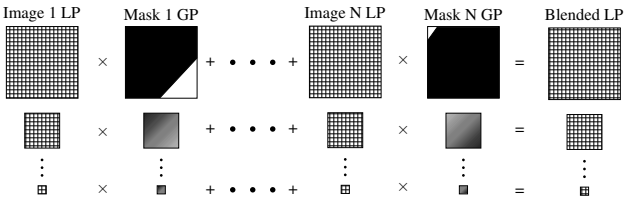

Fig. 3. Illustration of MBB

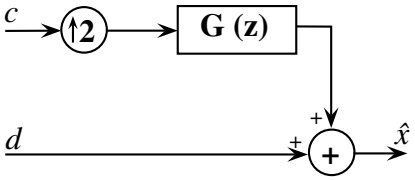

Fig. 4. Two-level LP reconstruction

column, pixelwise. The analogous implementation is also possible, where reading is done column-by-column, and the output pixels are provided row-by-row. However, most of the standard image resolutions have higher horizontal than vertical resolution. Hence, the first option is chosen, because the buffers in Figure 1 can be significantly smaller when storing two columns, instead of two rows. To cancel the edge effect at the image boundaries, the edge extension is performed by reflecting two rows (columns) across the edges.

Figure 2 shows the implementation of the filter block. It can be observed from (3) that pixels in the first and the fifth row of the window are multiplied with the same coefficients. Hence, the same hardware architecture can be used in both cases. The similar situation occurs with the second and the fourth row, with different coefficients. In Figure 2, signals p_1 - p_5 denote the intensities of pixels in the columns 1-5. Output signals r_1 - r_5 denote the values of the filtered rows. To obtain a final filtered value of the pixel, r_1 - r_5 are summed.

An important benefit of this implementation lies in the absence of any hardware multipliers. Multiplications by 2, 4 and 8 are realized by binary shifts to the left by 1, 2 and 3 bits, respectively. The only multiplicand which is not a power of 2 is 3 and it is obtained by adding the operand to its double, *i.e.* shift by 1.

The operating frequency of the system is increased by placing pipeline registers following each addition. The registers are, however, not needed after shift operations, since logical shifts do not require any processing time. The advantages of this pipeline architecture in terms of system performance is shown in Section 4.

3.2. Blending

In addition to LP decomposition, MBB requires a weight mask for each LP level of the image, as explained in Section 2. Weight mask GPs are pre-calculated and stored in a RAM, since their size is much smaller than the size of the original image. The first level of the pyramid is filled with only 1 and 0. Hence, if the image resolution is $K \times M$, the lowest level of the GP occupies $K \times M$ bits in the RAM. The weights in the second level of GP can be represented with 4 bits. Since the resolution of the second level is $\frac{K \times M}{4}$ pixels, the total bit size is the same as the first level. This rationale can be applied to all following levels. Hence, the total size that the GP occupies in RAM is $K \times M \times L$ bits, where L is number of levels in the pyramid.

Figure 3 illustrates the process of MBB. N images are simultaneously decomposed into their respective LPs in N parallel branches. The process of decomposition is synchronized in a manner to avoid storing the LPs in the RAM. Weights are read from the memory and multiplied by the corresponding coefficient in the LP. The weighted coefficients from each image, *i.e.* parallel branch, are summed to form a blended LP. The computation of the blended LP is also realized in a pipeline, with registers following each multiplier and adder.

Different LP levels cannot be simultaneously blended. The highest level, *i.e.* the one smallest in size, is blended later, since the LP level is obtained last. Hence, the blended LP levels are not obtained at the same time and they have to be stored in the RAM. These are the only intermediate results that have to be stored.

3.3. Laplacian Pyramid Reconstruction

Reconstructing the resulting LP is realized as shown in Figure 4. The LP coefficients are read from the RAM, starting from the highest level, *i.e.* low-pass image. The coefficients are read in the same manner as presented earlier, *i.e.* row-by-row. The same reconstruction filter $G(z)$ from (3) is used. The coarse image (c) is upsampled and interpolated to increase the image resolution. Afterwards, the detail image (d) is added. The resulting image ($\hat{x}$) is used as a coarse image input to the next level of reconstruction.

4. EXPERIMENTAL RESULTS

The hardware design is implemented on a Virtex-7 FPGA development board VC707, with 1 GB of external 800 MHz Dual Data Rate type 3 (DDR3) RAM. The maximum synthesizable operating clock frequency in the design is 420 MHz. The tightest constraint on the clock frequency is imposed by the adders in Figure 2. It is important to note that the proposed design is driven by only one clock signal. In [11] and [12], each pyramid level is driven by a different clock; each higher level in the pyramid is decomposed using a four times slower clock signal. Having only one clock domain is especially important if the design is to be fabricated as an ASIC, where clock routing becomes complicated in case of multiple clock trees.

The implemented design supports a dual video stream, but it can be extended to support more cameras, if it is needed. The data from the camera is recorder in RGB format, with 8 bits depth per color channel. An LP decomposition is done for each color channel, and they are operating in parallel. Each frame is decomposed into 4 LP levels, which is the maximum possible number for the chosen resolution. The display video resolution is 1920 × 1080 (HD 1080).

Table 1 shows the FPGA resource utilization summary and comparison to related work. In this work, the external RAM is only used

Table 1. FPGA resource usage comparison

	This work	**[11]**	**[12]**
Resource		**Used**	
Slices	7467	13287	2641
BlockRAM	14	430	38
DSP	4	–	–
External RAM [MB]	8.26	1.22	1.20
Family	Virtex-7	Virtex-2	Virtex-4

(a) (b)

Fig. 5. (a) Prealigned captured images and their masks; (b) photo-mosaic of EPFL campus created using the proposed MBB implementation

Table 2. Timing performance comparison

	This work	[11]	[12]	[13]
Max. freq. [MHz]	420	31	–	20
Frame rate [fps]	94	101	25	55
Resolution	HD 1080	VGA	VGA	512×512
Pyramid levels	4	4	3	10
Pixel depth [bits]	24	8	8	8

for storing mask GPs and the resulting LP. Larger external RAM occupancy is only due to the increased image resolution and color images. A reduction in BlockRAMs for temporary data storage and in used FPGA slices is observed. The design uses more slices than [12] because of the difference in filtering implementations, and one more LP decomposition level. In this work, filtering is realized using only adders made of Look-Up Tables (LUT) and registers, instead of multipliers in DSP blocks. The data from related work is taken from the original publications. Unknown information is represented by the "–" sign in all tables.

Table 2 shows the timing performance of blending two video streams. The maximum operating frequency is much higher than observed in the previous implementations. Apart from the newer FPGA family, the speed improvement is further influenced by a fully pipelined computation architecture. The achieved frame rate of 94 fps is very close to the best performance of the related systems [11]. However, the proposed design achieves this frame rate for significantly higher display resolution.

A comparison with GPU implementations is given in Table 3. The FPGA implementation is superior to the GPU. The GPU solutions are not able to achieve frame rates higher than 2 fps for display resolutions of more than 1 MP. Furthermore, the proposed architecture has a constant processing speed independent of the amount of overlap between the images. This is an important advantage compared to the possible software implementations.

Table 3. FPGA vs. GPU performance comparison

	This work	[9]	[10]
GPU	–	GeForce 8	Quadro 4600
Frame rate [fps]	94	0.43	1.79
Resolution	HD 1080	1147×608	1024×1024
Pyramid levels	4	–	7
Pixel depth [bits]	24	24	24

Figure 5 illustrates result of the proposed design. The images in Figure 5(a) are taken using two Aptina MT9P031 5MP sensors. The images are prealigned on a PC and stored into memory of VC707 Xilinx development board. The first level of the GP is also shown in Figure 5(a). The resulting blended image is shown in Figure 5(b).

5. CONCLUSION

In this paper, we propose a fast multiple-image blending hardware implementation. The blending algorithm is based on a multiresolution decomposition into an LP and image blending in different frequency bands. The proposed pipeline implementation is faster and less resource-demanding than the previous solutions. The experimental results show that the proposed implementation achieves higher or the same frame rates as the previously designed systems, but at a much higher, HD resolution. Furthermore, superiority of the design over GPU solutions, under similar benchmark tests, is shown in the comparison.

6. REFERENCES

[1] R. Szeliski and H-Y. Shum, "Creating Full View Panoramic Image Mosaics and Environment Maps," in *Proceedings of the Conference on Computer Graphics and Interactive Techniques*, New York, NY, USA, 1997, SIGGRAPH '97, pp. 251–258, ACM.

[2] S. E. Chen, "Quicktime VR: An Image-based Approach to Virtual Environment Navigation," in *Proceedings of the Conference on Computer Graphics and Interactive Techniques*, New York, NY, USA, 1995, SIGGRAPH '95, pp. 29–38, ACM.

[3] M. Brown and D. Lowe, "Automatic Panoramic Image Stitching Using Invariant Features," *International Journal of Computer Vision*, vol. 74, no. 1, pp. 59–73, August 2007.

[4] S. Peleg and J. Herman, "Panoramic Mosaics by Manifold Projection," in *IEEE Conference on Computer Vision and Pattern Recognition*, San Juan, Puerto Rico, June 1997, pp. 338 –343.

[5] P. Burt and E. Adelson, "A Multiresolution Spline with Application to Image Mosaics," *ACM Trans. Graph.*, vol. 2, no. 4, pp. 217–236, Oct. 1983.

[6] M-S. Su, W-L. Hwang, and K-Y. Cheng, "Variational Calculus Approach to Multiresolution Image Mosaic," in *Proceedings*

of International Conference on Image Processing, Oct. 2001, vol. 2, pp. 245 –248.

[7] B. Wilburn, N. Joshi, V. Vaish, E-V. Talvala, E. Antunez, A. Barth, A. Adams, M. Horowitz, and M. Levoy, "High Performance Imaging Using Large Camera Arrays," *ACM Trans. Graph.*, vol. 24, pp. 765–776, July 2005.

[8] H. Afshari, A. Akin, V. Popovic, A. Schmid, and Y. Leblebici, "Real-Time FPGA Implementation of Linear Blending Vision Reconstruction Algorithm Using a Spherical Light Field Camera," in *IEEE Workshop on Signal Processing Systems*, 2012.

[9] B. Daga, A. Bhute, and A. Ghatol, "Implementation of Parallel Image Processing Using NVIDIA GPU Framework," in *Proceedings of International Conference on Advances in Computing, Communication and Control*, Mumbai, India, January 2011, pp. 457–464.

[10] P. P. Shete, P. P. K. Venkat, and S. K. Bose, "Pyramidal Image Blending Using CUDA Framework," *International Journal of Engineering Science and Technology*, vol. 3, no. 12, pp. 8502–8513, December 2011.

[11] O. Sims and J. Irvine, "An FPGA Implementation of Pattern-Selective Pyramidal Image Fusion," in *International Conference on Field Programmable Logic and Applications*, August 2006, pp. 1–4.

[12] Y. Song, K. Gao, G. Ni, and R. Lu, "Implementation of real-time Laplacian pyramid image fusion processing based on FPGA," *Proceedings of SPIE*, vol. 6833, 2007.

[13] G. S. Van Der Wal and P. J. Burt, "A VLSI Pyramid Chip for Multiresolution Image Analysis," *International Journal of Computer Vision*, vol. 8, no. 3, pp. 177–189, Sept. 1992.

[14] R. Szeliski, *Computer Vision: Algorithms and Applications*, Springer, New York, NY, USA, 2011.