*IEIE Transactions on Smart Processing and Computing, vol. 3, no. 3, June 2014*
http://dx.doi.org/10.5573/IEIESPC.2014.3.3.110


# Depth-adaptive Sharpness Adjustments for Stereoscopic Perception Improvement and Hardware Implementation

**Hak Gu Kim, Jin Ku Kang, and Byung Cheol Song**

Department of Electronic Engineering, Inha University / Incheon, South Korea bcsong@inha.ac.kr

***** Corresponding Author: Byung Cheol Song

*Received November 22, 2013; Revised January 5, 2014, Accepted March 20, 2014; Published June 30, 2014*

** Regular Paper*

***Abstract*:** This paper reports a depth-adaptive sharpness adjustment algorithm for stereoscopic perception improvement, and presents its field-programmable gate array (FPGA) implementation results. The first step of the proposed algorithm was to estimate the depth information of an input stereo video on a block basis. Second, the objects in the input video were segmented according to their depths. Third, the sharpness of the foreground objects was enhanced and that of the background was maintained or weakened. This paper proposes a new sharpness enhancement algorithm to suppress visually annoying artifacts, such as jagging and halos. The simulation results show that the proposed algorithm can improve stereoscopic perception without intentional depth adjustments. In addition, the hardware architecture of the proposed algorithm was designed and implemented on a general-purpose FPGA board. Real-time processing for full high-definition stereo videos was accomplished using 30,278 look-up tables, 24,553 registers, and1,794,297 bits of memory at an operating frequency of 200MHz.

***Keywords*:** 3D, stereo video, sharpness enhancement, object, segmentation, FPGA

## 1. Introduction

As three-dimensional (3D) consumer electronics products, such as 3DTV, 3D monitors and 3D smart phones, have become increasingly popular, along with the success of 3D movies, more interest has been attracted to 3D image processing. Stereoscopic 3DTV can provide viewers with the impression of depth and a greater sense of presence [1]. The main depth cue used by the human visual system comes from the horizontal differences (parallax) between the two ocular viewpoints. To control stereoscopic perception in a 3D display, it is important to properly adjust the depth, or parallax. Many depth adjustment algorithms have been developed [2-8]. For example, some researchers, such as Kim and Sohn [2], proposed controlling the depth information based on visual fatigue [2-6]. Fig. 1 describes the general depth adjustment algorithm. First, the pixel distance between the left-eye view frame (LVF) and right-eye view frame (RVF) (i.e., disparity) was estimated. Note that the disparity actually corresponds to depth information. Second, the appropriate depth was selected by considering the visual fatigue level, or the stereoscopic perception. Third, the pixels in the LVF or RVF were moved according to the adjusted depth. In this step, so-called holes can occur around the shifted pixels. Finally, these holes were filled using appropriate interpolation methods [9].

The main drawback of such a simple parallax shifting method is that it can create losses in the image area due to unavoidable cropping at the screen edges that occurs when eliminating the unpaired points. In addition, all of these hole-filling techniques can lead to a range of distortions, which may be noticeable and visually annoying. In particular, artificially growing the depth may increase the visual fatigue level.

This paper presents a sharpness adjustment algorithm for artifact-free stereoscopic perception improvement without a deliberate depth adjustment for objects in a 3D video sequences. First, the disparity for each LVF/RVF pair was estimated on a block basis. Second, object segmentation was performed according to the disparity, so that the foreground and background objects were discriminated. Finally, the sharpness of the foreground object was enhanced, and that of the background objects was maintained or lessened. For this step, a new sharpness enhancement algorithm was presented that mitigates

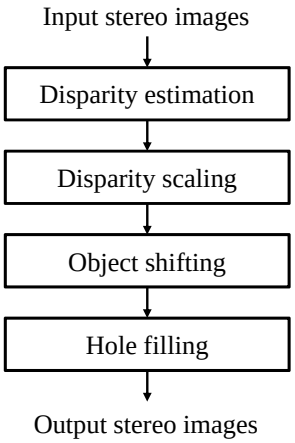



**Fig. 1. General depth adjustment method for stereoscopic video sequences.**

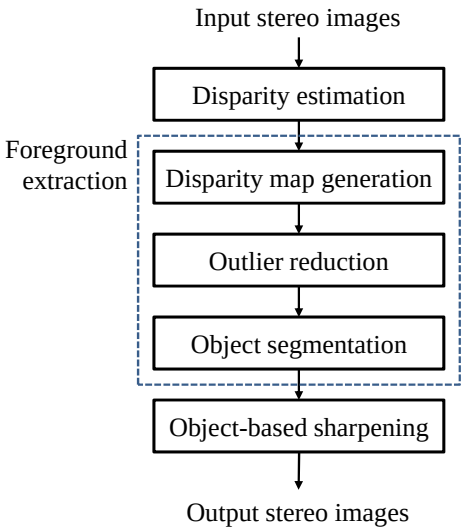



**Fig. 2. Overall flow of the proposed algorithm.**

jagging and halo artifacts. The experiment results showed that the proposed algorithm enhances depth perception without any visual fatigue caused by an artificial depth adjustment. In addition, the hardware architecture was designed for the proposed algorithm, and the results of its implementation are discussed. The proposed algorithm was created successfully on a dedicated 200MHz field-programmable gate array (FPGA) board operated in real time using the following resources: 30,278 look-up tables (LUTs), 24,553 registers, and 1,794,297 bits of memory. The implemented hardware will soon be applied to a specific 3D monitor product model.

The remainder of this paper is organized as follows: Section 2 describes the proposed algorithm. Section 3 reports the simulation results. Section 4 summarizes the very large scale implementation results, and Section 5 reports the conclusions.

## 2. The Proposed Algorithm

Fig. 2 presents the overall flow of the proposed algorithm. The main contribution points of the proposed algorithm are simple foreground extraction and edge-preserving sharpness enhancement. The following subsections depict each step of the proposed algorithm in detail.

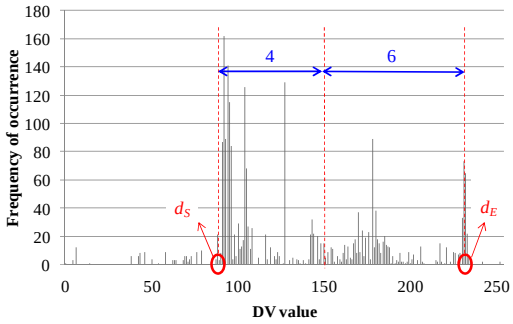
### 2.1 Disparity Estimation

For several decades, many disparity estimation algorithms for stereo images have been developed [10-14]. To enable real-time hardware implementation, this paper adopts a typical block-matching-based disparity estimation algorithm because of its low computational complexity. Note that block matching is performed only in the horizontal direction because it is assumed that the input stereo frames are already rectified. The matching block size was set to 16×16, and the searched disparity vector (DV) was assigned to the central 8×8 of the matching block via proper overlapping with its neighboring matching blocks. A typical three-step hierarchical search was used for further computational reduction. Levels 2, 1 and 0 represent the coarsest, middle, and finest resolutions, respectively. Prior to the search, the frames at the middle and coarsest resolutions, i.e., levels 1 and 2 (named $I^{(1)}$ and $I^{(2)}$, respectively) are produced by down-sampling the original frame by 1/2 and 1/4, respectively, in both directions. First, a level 2 search is performed on the 8×8 block. The best DV at level 2 ($\hat{d}^{(2)}$) is obtained by minimizing the sum of absolute differences (SAD) as follows:

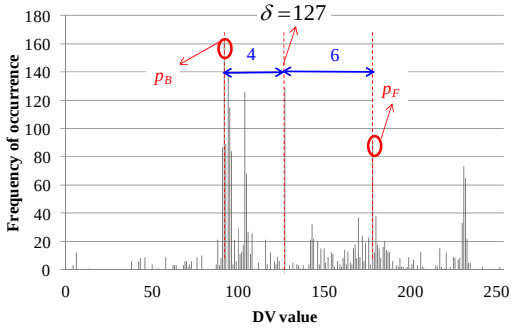
$$\hat{d}^{(2)} = \arg\min_{d \in \Omega^{(2)}} \sum_{(i,j) \in W^{(2)}} \left| I_L^{(2)}(i,j) \cdot I_R^{(2)}(i+d,j) \right| \qquad (1)$$

where $I_L^{(2)}$ and $I_R^{(2)}$ represent the LVF and RVF, respectively, at level 2, and (i, j) denotes the coordinate of the upper-left corner pixel in matching block $W^{(2)}$. The search range at level 2, $\Omega^{(2)}$ is set to [-32, 32]. Similarly, the search at level 1 is performed for a local search area with the center being 2x $\hat{d}^{(2)}$, and the best DV at level 1, $\hat{d}^{(1)}$, is found. The matching block size at level 1 is 8×8, and $\Omega^{(1)}$ is set to [-1, +1]. Finally, the search at level 0 is performed for a local search area $\Omega^{(0)}$ with the center being 2× $\hat{d}^{(1)}$, and the best DV $\hat{d}$, $\hat{d}^{(0)}$, is found. The matching block size is 16×16, and $\Omega^{(0)}$ is set to [-1, +1].

After obtaining the DV for each block, the so-called bi-directional check [15] is performed to investigate the accuracy of block matching. That is, if the target block in the LVF is matched to a particular block in the RVF with the corresponding DV $\hat{d}$, the best DV of the matched block in the RVF is explored in a given search area in the LVF. If such a reverse DV estimation is accomplished, whether or not the reverse DV is - $\hat{d}$ can be determined. If it is, $\hat{d}$ is determined to be reliable. Otherwise, the DV of




(a)




(b)

**Fig. 3. DV histogram-based object segmentation (a) the start and end points, (b) threshold determination between the object and background.**

the target block is replaced with the median of the DVs of the neighbor blocks because $\hat{d}$ may be unreliable. In addition, a morphological closing operation and median filtering is applied to the derived DV map on a 3×3 block basis. In this manner, a block-based disparity map is obtained.

## 2.2 Disparity-based Foreground Segmentation

The second step in the proposed algorithm is to extract the dominant foreground object(s) using the DV histogram. Fig. 3 illustrates the extraction process from a typical DV histogram. One feature of the DV histogram is that the DVs of a foreground object are generally located on the right side of the DV histogram; those of the background object are located on the left side. Another feature of the DV histogram is that the DVs of the background object(s) tend to be similar and gathered together. Based on these two features, an object segmentation is performed on the DV histogram. First, the start point and end point having meaningful non-zero bin-values ( $d_S$ and $d_E$ ) are found on the DV histogram, as shown Fig. 3(a). For example, $d_S$, which meets the following condition (2), is found, searching from the left using:

$$\sum_{k=0}^{4} H(d_S + k) \geq \theta \qquad (2)$$

where $H(t)$ denotes a histogram bin-value of disparity $t$ and the threshold $\theta$ is fixed empirically to 3% of the total sum of disparity bin-values. Similarly, $d_E$, which meets the similar condition, is found starting from the right, respectively. If $d_S$ and $d_E$ are determined, then the foreground and background are extracted. [ $d_S$ , $d_E$ ] are divided empirically into two clusters in terms of the $L_1$-norm distance at a ratio of 4:6. The left group and right group are initially regarded as the background and foreground, respectively. Each peak, $p_F$ and $p_B$, is found in each group for more precise foreground and background segmentation. $p_F$ and $p_B$ are the peaks in the foreground group and background group, respectively. After finding the two peaks, the new determination value, $\delta$, is updated, as shown Fig. 3(b). Finally, a labeling operation is applied to only the foreground pixels. A well-known connected-component labeling algorithm is adopted [16]. After labeling, several foreground objects can be produced. In this paper, the largest object was adopted as the most dominant foreground object.

## 2.3 Object-based Sharpness Enhancement

The last step of the proposed algorithm is to enhance the sharpness of only the selected foreground object. In this study, the remaining regions in the frame were maintained without processing. Note that a typical sharpness enhancement algorithm may cause unwanted artifacts, such as jagging, the halo effect, and noise boosting. Polesel et al. employed an adaptive filter that controls the sharpness in such a way that contrast enhancement occurs in high-detail areas with little or no image sharpening occurring in the smooth areas [17]. The adaptive filter emphasizes the medium-contrast details in the input image more than the large-contrast details, such as abrupt edges, to avoid overshoot effects in the output image. Therefore, the adaptive unsharp masking (AUM) method first divides each input image into three regions: smooth, medium-contrast, and high-contrast regions. The adaptive filter does not perform a sharpening operation in smooth areas. Therefore, the overall system is more robust to the presence of noise in the input images than the traditional approaches. In addition, the local dynamics in the high-contrast areas are already high, and such regions require only moderate sharpening. The medium-activity areas require the most enhancements. Based on this, the AUM applies strong sharpening to the medium-contrast regions, whereas moderate weak sharpening is applied to the high-contrast regions. In this manner, the AUM accomplishes the dual objectives of avoiding noise amplification and excessive overshoot in the detail areas. According to the experimental results in Section 3, the AUM algorithm does not resolve jagging or halo artifacts.

This paper proposes a sharpness enhancement algorithm that provides less computational complexity and fewer artifacts than the AUM algorithm. Fig. 4 describes the proposed algorithm. In this figure, $x(m,n)$ and

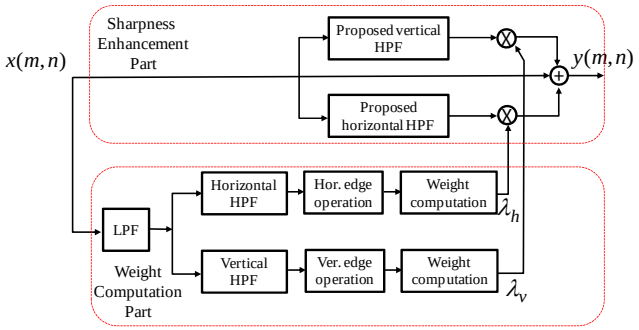



**Fig. 4. Proposed sharpness enhancement algorithm.**

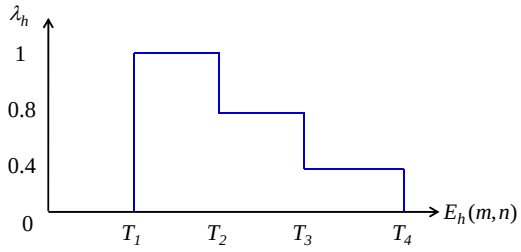



**Fig. 5. Adaptive weighting.**

$y(m,n)$ indicate a pixel located at $(m,n)$ in the LVF or RVF input, respectively, and the processed output. Like the AUM, the proposed algorithm consists of two parts: pixel-wise weight computation and sharpness enhancement. The details are described in the following subsection.

### 2.3.1 Weight Computation according to Edge level

For convenience, only the weight computation process for the horizontal direction is discussed. First, a typical low-pass filter (LPF) and high-pass filter (HPF) are applied sequentially to $x(m,n)$. Second, the horizontal edge level of the processed pixel, i.e., $\varepsilon_h(m,n)$, is computed, where the Sobel edge operator is used and the horizontal edge level means the absolute value of the magnitude of the horizontal edge obtained by Sobel operator on the horizontal direction. Finally, the horizontal weight $\lambda_h$ is determined adaptively according to the edge level, $\varepsilon_h(m,n)$, as shown in Fig. 5. In the case of a weak edge area, when $\varepsilon_h(m,n) < T_1$, $\lambda_h$ is set to 0, because it requires only a slight sharpening effect. For a medium-activity area, which mostly affects the human visual system, where $T_1 \le \varepsilon_h(m,n) < T_2$, $\lambda_h$ is set to 1. Furthermore, $\lambda_h$ decreases with increasing $\varepsilon_h(m,n)$. In this study, $\lambda_h$ was determined for the five intervals of $\varepsilon_h(m,n)$. For extremely high activity areas, when $\varepsilon_h(m,n) \ge T_4$, $\lambda_h$ was fixed at 0 to avoid the overshoot artifacts. The weights and thresholds were determined empirically according to intensive experiments for various stereo video sequences. As a result, in this study, $T_1$, $T_2$, $T_3$, and $T_4$ were set to 20, 40, 70, and 100, respectively.

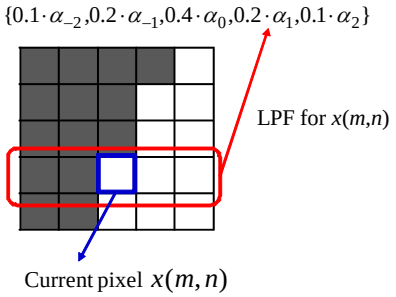



**Fig. 6. Proposed edge-preserving LPF in the horizontal direction.**

### 2.3.2 Sharpness Enhancement

After the horizontal and vertical weights, i.e., $\lambda_h$ and $\lambda_v$, are obtained, the sharpness enhancement using these weights is applied to $x(m,n)$, as shown in the upper part of Fig. 4. To mitigate the halo or jagging artifacts, this paper proposes the use of an edge-preserving LPF. As seen in Fig. 4, the sharpness enhancement was performed using one-dimensional processing to enable simple hardware implementation. Fig. 6 illustrates the basic concept of the proposed edge-preserving LPF. The LPF coefficients for the five rows in the 5×5 filtering processing block were determined adaptively. The proper weights $\{\alpha_{-2},\alpha_{-1},\alpha_0,\alpha_1,\alpha_2\}$ are multiplied into a typical 5-tap Gaussian LPF, i.e., {0.1, 0.2, 0.4, 0.2, 0.1}. The weights were computed according to:

$$\alpha_k = \begin{cases} 1 & if\ |x(m,n)\cdot x(m,n+k)| < T_S \\ 0 & otherwise \end{cases} \quad (3)$$

In (3), the threshold $T_S$ was set to 60. In other words, a pixel that was significantly different from the current pixel $x(m,n)$ was excluded from the computation. Note that proper normalization follows. Using the computed LPF coefficients, $x(m,n)$ was low-pass-filtered. The edge-preserving HPF was then accomplished by subtracting the low-pass-filtered $x(m,n)$ from the original $x(m,n)$. The proposed HPF provided clearer edges than the typical HPF. As a result, the halo effect around the strong edges was avoided. $\lambda_h$ was then multiplied with the horizontally HPF-ed output, and $\lambda_v$ was similarly multiplied with the vertically HPF-ed output. Finally, $y(m,n)$ was obtained by adding the results to $x(m,n)$.

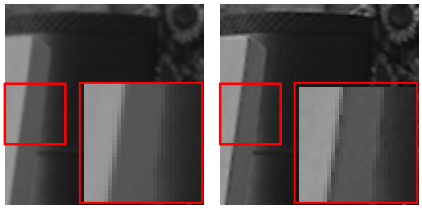
## 3. Performance Evaluation

Four Middlebury stereo images were used to evaluate the performance of the proposed algorithm; *Reindeer*, *Cones*, *Dwarves*, and *Art*. The frame size of all of the test sequences was 1920×1080. The frame format was side-by-

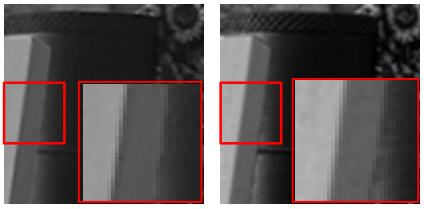
(a)

(b)

**Fig. 7. Segmentation of foreground areas for *Reindeer* (a) the input LVF/RVF, (b) the output LVF/RVF.**



(a) (b)



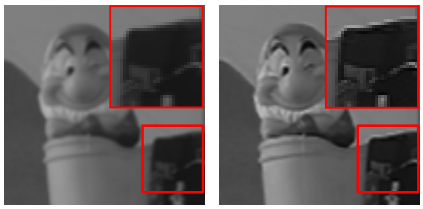
(c) (d)

**Fig. 8. Sharpness enhancement results for *Dwarves* (a) original, (b) AUM algorithm, (c) proposed algorithm, (d) result of GUM [19]. Here a specific part of the image is cropped.**

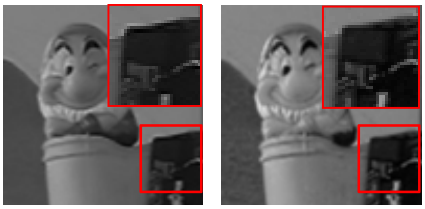
side. Therefore, in the case of the left–right (L/R) side-by-side format, the frame size of the LVF/RVF was 960×1080, and in the case of the top–bottom (T/B) side-by-side format, the frame size of the LVF/RVF was 1920×540.

Fig. 7 shows the foreground extraction result for the *Reindeer* image in T/B format. To avoid visually annoying artifacts around the object boundary when enhancing the sharpness, an 8-pixel exterior band around the boundary of the foreground object was also assigned to the foreground object.

For the comparisons, the AUM and generalized unsharp masking algorithm (GUM) were employed [19]. Segments of the results are displayed in Fig. 8. The AUM results suffered from jagging artifacts around the strong



(a) (b)



(c) (d)

**Fig. 9. Sharpness enhancement results for *Dwarves* (a) original, (b) the algorithm, (c) proposed algorithm, (d) result of GUM [19]. Here a specific part of the image is cropped.**

(a) (b)

**Fig. 10.Example of the proposed algorithm results for *Dwarves* (a) the input stereo image, (b) output stereo image.**

(a) (b)

**Fig. 11. Example of the proposed algorithm results for *Art* (a) input stereo image, (b) output stereo image.**

edges (see Fig. 8(b)). In addition, the GUM results provided weak sharpness on the bright intensity range in Fig. 8(d). In contrast, the proposed algorithm mitigated these jagging artifacts while maintaining the sharpness around the edges, as shown in Fig. 8(c). Fig. 9 presents another result. The halo effect can be seen around the patterns in Fig. 9(b) and jagging artifacts can be observed around the diagonal edges. The GUM and proposed algorithm significantly suppressed such phenomena while enhancing the sharpness, but GUM boosted the noise at the flat areas as shown Fig. 9(d).

Figs. 10 and 11 compare the final outputs from the

**Table 1. Comparison results for various test images based on MSSSIM.**

| Images | AUM | GUM | Proposed |
|---|---|---|---|
| Reindeer | 0.9854 | 0.8951 | 0.9865 |
| Cones | 0.9809 | 0.9633 | 0.9873 |
| Dwarves | 0.9920 | 0.9485 | 0.9922 |
| Art | 0.9865 | 0.7064 | 0.9874 |
| Average | 0.9862 | 0.8783 | 0.9884 |

*Dwarves* and *Art* images in the anaglyph, respectively. The foreground objects (e.g., the dwarves and the plaster cast) were sharpened except for background wallpaper. On a typical 3D monitor, the stereo video sequences manipulated by the proposed algorithm were found to provide better depth perception than the original video sequences, and reduced visual fatigue. To apply the proposed algorithm to a particular 3D monitor, it was implemented on an FPGA platform, which will be described in the following section.

For comparison in terms of the objective visual quality, an image quality assessment metric for an objective evaluation of the sharpness enhancement: multi-scale structural similarity (MSSSIM) can be employed [18].

Table 1 lists the MSSSIM values for various algorithms. Noteworthy is that the closer the MSSSIM is to 1, the closer the sharpness of the test image to that of the original. As shown in Table 1, the proposed algorithm provides higher MSSIM values than the AUM and GUM on average.

## 4. Hardware Implementation

### 4.1 Architecture Design

The target video has a resolution of 1920x1080 and a frame rate of 60Hz. A careful design is needed to process, such a full high-definition stereo video in real time. Dual buffering of the frame units, parallel connection of the random access memory (RAM) in the FPGA, and pipelining modules were all applied in the hardware design for real-time operation.

The hardware was designed to operate at 200 MHz for real-time operation on a dedicated FPGA board. For the 200MHz operation, the clock cycles required for the one matching block (MB) calculation were determined by:

$$MB\,per\,Sec = \frac{1920\times1080\times60(pixel/s)}{256(pixel/MB)} = 486\times10^{3}(MB/s)$$

$$clock\,cycle\,per\,MB = \frac{200\times10^{6}(cycle/s)}{486\times10^{3}(MB/s)} = 412(cycle/MB) \quad (4)$$

Therefore, approximately 49,440 clock cycles are required for a one slice calculation because there are 60 MBs in a single slice. In the disparity estimation at level 2, which has the heaviest calculation load, the number of pixels to be read was 640 (8×8 pixels from the right side, 72×8 pixels from the left side). If one pixel is read in each clock cycle, 640 clock cycles are required, which is more than the allocated cycles (412 cycles) in a single MB calculation for 200MHz operation. Because 1,300 cycles are needed to handle one MB, and one slice has 60 MBs, a total of 78,000 clock cycles will be needed without pipelining for a single slice-disparity estimation, which is also more than the 49,440 cycles allowed by the 200MHz operation. To reduce the number of required clock cycles to conform 200MHz operation, 16 parallel pixels were first read together from the line buffers in a single cycle. Therefore, the clock cycles for reading the data from both sides of the line buffers for the level 2 disparity estimation were reduced by 36 cycles from the 640 cycles. Furthermore, by pipelining the disparity estimation process, the clock cycles for a single slice calculation are reduced by 5,300 from 78,000 cycles. The number of clock cycles required for the later processes, such as filtering, histogram building, and labeling, was approximately 3,740 cycles. Approximately 20,000 clock cycles were consumed while waiting to fill the line buffers. Therefore, the total number of cycles used in this implementation was approximately 29,040, which is less than the 49,440 cycles available for the 200MHz operation.

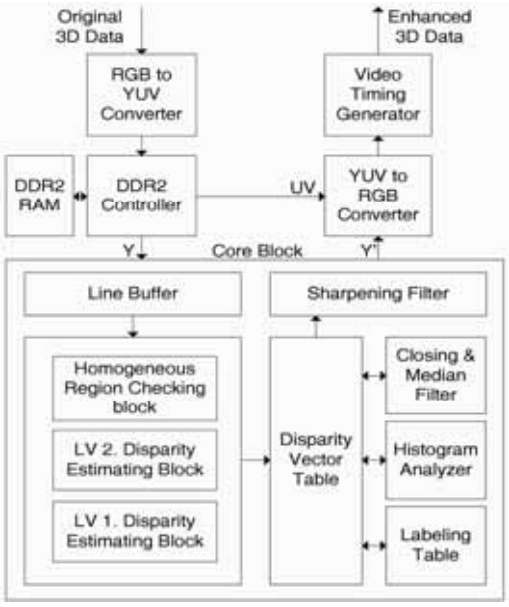



**Fig. 12. Hardware block diagram.**

Fig. 12 presents a block diagram of the hardware implementation for the algorithm. The hardware consists of 13 modules. A 1GB module of DDR2-800 SO-DIMM RAM is used as the external RAM, which is controlled by the high-performance memory controller embedded in the FPGA. The pixel data, converted from red-green-blue (RGB) 4:4:4 to luminance-bandwidth-chrominance (YUV) 4:4:4, is read in real time at 60Hz. The line buffer stores the left and right side pixel data slices for the search operation. One slice consists of 32 horizontal lines, but only 16 lines are stored in the line buffer because there is no disparity estimation in level 0.

When both line buffers are full, the disparity estimation

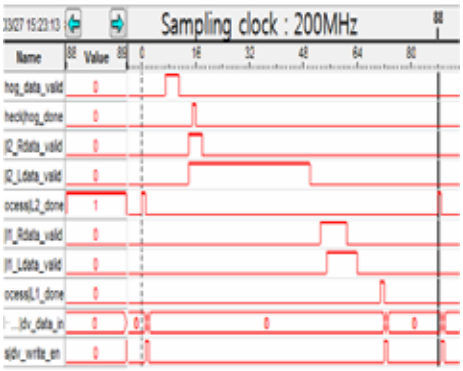



**Fig. 13. Waveform for one matching block calculation.**

**Table 2. Summary of the FPGA synthesis results.**

| | | |
|---|---|---|
| Logic Utilization | LUTs | 30,278 (17%) |
| | Registers | 24,553 (13%) |
| Total Registers | | 24,555 |
| Total Block Memory bits | | 1,794,297 |

block begins to calculate the DV values. The core block extracts the foreground region using the proposed algorithm. The foreground region information is stored in internal RAM. The stored data is then read out from RAM, and the sharpening is processed.

For the RVF data, the stored foreground region determines whether to proceed with the sharpening. Because there is only a position difference between the LVF and RVF data, the shifted foreground region can be used for LVF sharpening. Therefore, the foreground extraction for the LVF can be omitted.

If the data is determined to be a foreground, the luminance (Y) data from the sharpening filter and the chroma (UV) data from external RAM are combined for the final YUV data. If the data is determined to be the background, the UV data and Y data without sharpening are combined for the final YUV data. The final YUV data are converted back to RGB for an enhanced 3D data display.

## 4.2 FPGA Implementation

The proposed hardware architecture was coded in Verilog HDL and implemented with a dedicated FPGA. The implementation was verified using RTL simulations. The HDL model was synthesized on the FPGA device. Fig. 13 shows a simulation waveform of the process used to calculate the DV of a MB, as an example of the implementation. The waveform shows the internal signals of the FPGA operating at 200MHz. This also shows that a total of 88 clock cycles are needed for one MB disparity estimation. Therefore, a total of 5,280 clock cycles were required for the 60 MBs disparity estimation process. This is similar to the estimated cycles (5,300 cycles) described in Section 4.1.

Table 2 lists the synthesis results. This shows that 30,278 LUTs, 24,553 registers, and 1,794,297 bits of memory were used. For the real time processing of the algorithm for a 1080p 3D display, the core block was designed to operate at 200MHz. The maximum operating frequency was measured at 213.36MHz. The other blocks, such as the display control and the RGB to YUV conversion, were set to 148.35MHz for the connected displays.


**Fig. 14. Example of a typical 3D image with the algorithm implemented on an FPGA board.**

As shown in Fig. 14, the functionality of the proposed algorithm was verified using a typical 3D image with the algorithm implemented on the FPGA board.

## 5. Conclusions

This paper proposed a sharpness adjustment algorithm that provides artifact-free stereoscopic perception improvement for 3D video sequences. The proposed algorithm was shown empirically to improve stereoscopic perception without visual fatigue because there is no intentional disparity adjustment for objects. In addition, the hardware architecture of the proposed algorithm was designed, and the real-time processing for full HD stereo videos was demonstrated on a general-purpose FPGA development board with an operating frequency of 200MHz, using 30,278 look-up-tables, 24,553 registers, and 1,794,297 bits of memory. As a result, the proposed framework can be a possible solution for artifact-free stereoscopic perception improvement for 3D applications.

## References

[1] R.Hartley andA. Zisserman, Multiple View Geometry in Computer Vision, Cambridge University Press, 2000. Article (CrossRef Link)

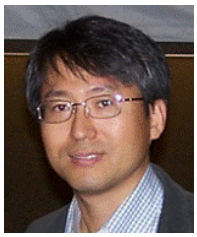
[2] D. Kim and W. Sohn, “Depth adjustment for stereoscopic image using visual fatigue prediction and depth-based view synthesis,” Proc. Of IEEE International Conference on Multimedia and Expo, pp. 956-961, Jul. 2010. Article (CrossRef Link)

[3] S. Kishi, S. H. Kim, T. Shibata, T.Kawai, J. Hakkinen, J. Takatalo, and G. Nyman, “Scalable 3D image conversion and ergonomic evaluation,” Proc. SPIE, vol. 6803, 2008. Article (CrossRef Link)

[4] J. Konrad, B. Lacotte and E. Dubois, “Cancellation of image crosstalk in time sequential displays of

stereoscopic video," IEEE Trans. Image Processing, vol. 9, no. 5, pp. 897-908, 2000. Article (CrossRef Link)
[5] J. Park, G. Um, C. Ahn, and C.-T. Ahn, "Virtual control of optical axis of the 3DTV camera for reducing visual fatigue in stereoscopic 3DTV," ETRI Journal, pp. 597-604, 2004. Article (CrossRef Link)
[6] N. Holliman, "Mapping perceived depth to regions of interest in stereoscopic images," in Proc. SPIE, Stereoscopic Disp. Virtual Reality Syst. XI, 2004, vol. 5291, pp. 1-12. Article (CrossRef Link)
[7] G. Sun and N. S. Holliman, "Evaluating methods for controlling depth perception in stereoscopic cinematography," in Proc. Stereoscopic Displays Virtual Reality Syst. XX, 2009, vol. 7237, pp.72370I-1-72370I-12. Article (CrossRef Link)
[8] S. J. Daly, R. T. Held, and D. M. Hoffman, "Perceptual issues in stereoscopic signal processing," IEEE Trans. Broadcast., vol. 57, no. 2, pp. 347-361, Jun. 2011. Article (CrossRef Link)
[9] A. Criminisi, P. Pérez, and K. Toyama, "Object removal by exemplar-based inpainting," in Proc. Conf. Computer Vision and Pattern Recognition, Madison, WI, Jun. 2003. Article (CrossRef Link)
[10] H. Kim, D. B. Min, S. Choi, and K. Sohn, "Real-time disparity estimation using foreground segmentation for stereo sequences," Optical Engineering, vol. 45, no. 3, pp. 037402(10pages) Mar. 2006. Article (CrossRef Link)
[11] S. H. Lee and S. Sharma, "Real-time Disparity estimation algorithm for Stereo Camera Systems," IEEE Transaction on Consumer Electronics, vol. 57, pp. 1018-1026, 2011. Article (CrossRef Link)
[12] D. Tzovaras, M. G. Strintzis, and H. Sahinoglou, "Evaluation of multi resolution block matching techniques for motion and disparity estimation," Signal Processing: Image Communication, vol. 6, no. 1, pp. 59-67, Mar. 1994. Article (CrossRef Link)
[13] L. Zhang, "Hierarchical block-based disparity estimation using mean absolute difference and dynamic programming," in Proc. Int. Workshop VeryLow Bit rate Video Coding, 2001, pp. 114-118. Article (CrossRef Link)
[14] E. Lee and Y. Ho, "Generation of multi-view video using a fusion camera system for 3D displays," IEEE Trans. On Consumer Electronics, vol. 56, no. 4, pp. 2797-2805, Dec. 2010. Article (CrossRef Link)
[15] M. EbroulIzuierdo, "Stereo image analysis for multi-viewpoint telepresence applications," Signal Processing: Image Communication, vol. 11, pp. 231-254, 1998. Article (CrossRef Link)
[16] L. Shapiro, and G. Stockman, Computer Vision, Prentice Hall, pp. 69-73. 2002. Article (CrossRef Link)
[17] A. Polesel, G. Ramponi, and V. Mathews, "Image enhancement via adaptive unsharp masking," IEEE Trans. Image Process., vol. 9, no. 3, pp. 505-510, Mar. 2000. Article (CrossRef Link)
[18] Z. Wang, E. P. Simoncelli, and A. C Bovik, "Multi-scale structural similarity for image quality assessment," presented at the IEEE Asilomar Conf. Signals, Systems, and Computers, Nov. 2003. Article (CrossRef Link)
[19] G. Deng, "A generalized unsharp masking algorithm," IEEE Trans. Image Process., vol. 20, no. 5, pp. 1249–1261, May 2011. Article (CrossRef Link)

**Hak Gu Kim** received his B.S. and M.S. degrees in electronic engineering from Inha University, Incheon, Korea in 2012 and 2014, respectively. Currently, he is pursuing a Ph.D. degree in electrical engineering from Korea Advanced Institute of Science and Technology (KAIST). His research interests include image processing, super-resolution, and video coding.



**Jin-Ku Kang** received his Ph.D. in electrical engineering from North Carolina State University in 1996. From 1983 to 1988, he worked at Samsung Electronics, Inc., Korea. From 1996 to 1997, he was with Intel as a senior design engineer. Since 1997, he has been a professor in the Department of Electronic Engineering, Inha University, Rep. of Korea. His research interests include high-speed CMOS VLSI design, mixed-mode IC design, and high-speed serial interface design.

**Byung Cheol Song** received his B.S., M.S., and Ph.D. degrees in electrical engineering from the Korea Advanced Institute of Science and Technology (KAIST), Daejeon, Korea, in 1994, 1996, and 2001, respectively. From 2001 to 2008, he was a senior engineer at Digital Media R&D Center, Samsung Electronics Co., Ltd., Suwon, Korea. In March 2008, he joined the Department of Electronic Engineering, Inha University, Incheon, Korea, and currently is an associate professor. His research interests are in the general areas of image/video processing, super-resolution, 3D vision, multimedia system design, and content-based multimedia retrieval.