A Real-Time High-Quality Complete System for Depth Image-Based Rendering on FPGA

Yanzhe Li, Luc Claesen, *Senior Member, IEEE,* Kai Huang, and Menglian Zhao

***Abstract*—Depth image-based rendering (DIBR) techniques have drawn more attention in various three dimensional (3D) applications nowadays. In this paper, a real-time high-quality DIBR system which consists of disparity estimation and view synthesis is proposed. For disparity estimation, a local approach that focuses on depth discontinuities and disparity smoothness is presented to improve the disparity accuracy. For view synthesis, a method that contains view interpolation and extrapolation is proposed to render high-quality virtual views. Moreover, the system is designed with an optimized parallelism scheme to achieve a high throughput, and can be scaled up easily. It is implemented on an Altera Stratix IV FPGA at a processing speed of 45 frames per second (fps) for 1080p resolution. Evaluated on selected image sets of the Middlebury benchmark, the average error rate of the disparity maps is 6.02%; the average peak signal to noise ratio (PSNR) and structural similarity (SSIM) values of the virtual views are 30.07 dB and 0.9303, respectively. The experimental results indicate that the proposed DIBR system has the top-performing processing speed and its accuracy performance is among the best of state-of-the-art hardware implementations.**

I. Introduction

EXTENDING visual sensation to 3D vision has been studied for decades and it is widely used in many applications, such as three dimensional television (3D-TV) and virtual reality gaming. They provide the audience with a greater sense of presence in a computer-generated environment. Specific viewing technology is used to ensure that each eye only observes one image from different viewpoints. However, the requirement to wear additional eyeglasses is usually perceived uncomfortable. Recently, autostereoscopic displays support glasses-free 3D depth perception by showing multiple views simultaneously so that the audience always sees a stereo pair from predefined viewpoints regardless of its position [1]. However, it is impractical to capture such multiple views by using a high number of cameras. The large amount of consecutive views, which are arranged properly to alleviate the motion parallax conflict, have significantly increased the amount of data to be processed and transmitted. Consequently, depth image-based rendering (DIBR), which efficiently generates novel realistic viewpoints using the known original views and the depth information, has been treated as a key part of 3D display systems [2].

In common DIBR algorithms, the production of high-quality virtual views not only poses challenges to the accuracy of the depth maps, but also requires additional inpainting steps to fill in the occluded regions that become visible in the synthesized views. It is noted that DIBR is a complicated and time-consuming procedure, which makes it hard to process in real time on a CPU. As a result, hardware acceleration is inevitable and it has been done extensively using GPUs and dedicated hardware. Current research efforts on DIBR focus on the high-quality performance and real-time processing speed in various 3D applications.

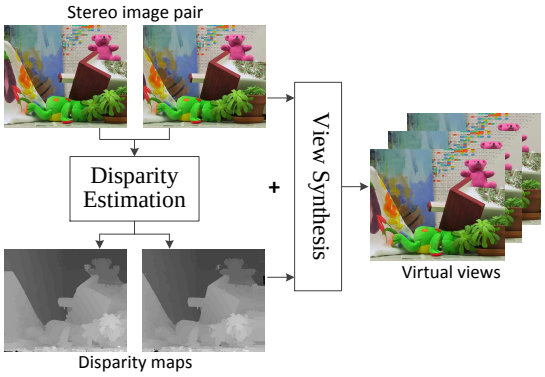


Fig. 1. The processing flow of the proposed DIBR algorithm.

DIBR algorithms can be mainly classified into two kinds: single-view approach and multi-view approach. In single-view approaches, depth information is generated from a single view based on the characteristics of the human depth perception. It is difficult to recover accurate physical depth even with high complexity algorithms [3]. Besides, another drawback is that the virtual views suffer from large occluded holes due to the lack of corresponding information of the background. On the contrary, in multi-view approaches, accurate depth maps are retrieved from the disparity of the correspondences in relative stereo/multiple images or videos. And the virtual views have smaller native holes because they can be filled using multi-view inputs.

In this paper, a stereo-view DIBR algorithm is proposed, which consists of two main parts: disparity estimation and view synthesis. The processing flow of the DIBR algorithm is illustrated in Fig. 1. In addition, a hardware system is presented based on the proposed DIBR algorithm, and the evaluation of our design is carried out on the Middlebury benchmark [4]. There are three major contributions for this paper:

1) A complete hardware-compatible DIBR algorithm which consists of disparity estimation and view synthesis is proposed for high synthesis quality and real-time pro-

Y. Li, K. Huang and M. Zhao are with the Institute of VLSI Design, Zhejiang University, Hangzhou 310058, China (e-mail: {liyz, huangk, zhaoml}@vlsi.zju.edu.cn).

L. Claesen is with the Engineering Technology - Electronics-ICT Dept., Hasselt University, Belgium (e-mail: luc.claesen@uhasselt.be).

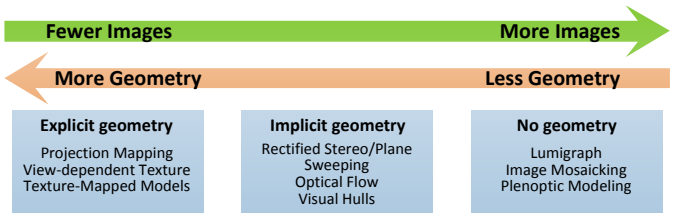


Fig. 2. Spectrum of the IBR classification.

cessing speed at high resolutions.

2) A sophisticated hardware architecture is presented with an optimized parallelism scheme to accelerate the proposed algorithm. It is designed to achieve a high throughput, and can be scaled up easily.
3) A prototype of the proposed DIBR system is developed on an Altera EP4SGX530 FPGA, which achieves a processing speed of 45 frames per second (fps) for 1080p resolution. Evaluated on selected image sets of the Middlebury benchmark, the average error rate of the disparity maps is 6.02%; the average peak signal to noise ratio (PSNR) and structural similarity (SSIM) values of the virtual views are 30.07 dB and 0.9303, respectively.

The rest of this paper is organized as follows. The background and some related works are introduced in Section II. The DIBR algorithm and its hardware implementation on an FPGA are described in Sections III and IV, respectively. The processing speed and accuracy performance of the DIBR system are discussed in Section V. Finally, this paper is concluded in Section VI.

II. Background and Related Work

A. Image-Based Rendering Classification

Conventional computer graphics requires an a priori specified 3D geometric model of the scene, together with color, texture, lighting, etc. The data is then processed by the rendering pipeline which performs projection of the 3D model to produce 2D images. Image-Based Rendering (IBR), on the other hand, generates virtual views directly from real-world captured images without full 3D model reconstruction. It takes advantage of the built-in degree of lifelike visual realism, which is difficult to achieve with traditional geometric modeling.

Although many IBR methods often defy rigid classification, they can still be broadly categorized on a spectrum, as shown in Fig. 2. It requires explicitly provided geometry on one end and little to no geometry on the other extreme, with implicitly determined geometry in between. A tradeoff is usually observed between the geometry and the number of images: the less geometry, the more input images are required, and vice versa [5].

IBR with implicit geometry first estimates the depth of the scene from any information contained in the input images. This kind of IBR algorithms is therefore regarded as DIBR. Disparity estimation from stereo images is a common approach to get depth information. It is also known as stereo matching, which takes a pair of rectified images, estimates the displacement of each pixel between the two images, and displays the associated movement in disparity maps. While disparity is often treated as synonymous with inverse depth, virtual views can then be synthesized by using the disparity maps.

B. Real-Time Hardware Systems

As mentioned in Section I, it is hard to process the complete DIBR algorithm in real time on a CPU. A CPU-based method to synthesize 3D video from a binocular stereo camera is presented in [6], achieving only 8 fps for 640 $\times$ 480 pixel images. To obtain real-time processing speed at high resolutions, GPUs are used as acceleration platforms. One of the first complete GPU-based systems for multi-view synthesis is proposed in [7], which runs at 24 fps using a pair of 1080p images. An L-shaped trifocal system which enables accurate scene reconstruction is proposed in [8]. High-quality 3D content is produced due to the spatial camera arrangement, but no real-time processing results are provided. A unified DIBR framework is proposed in [9] and implemented on GPUs at a frame rate of 10 fps with a 800 $\times$ 600 resolution. Although GPUs have proved to be an attractive speedup platform for DIBR-based systems, high power consumption restricts their performance.

Compared to GPUs, FPGAs have two advantages: 1) reconfigurable processing units and customized memory hierarchies and 2) low power. In most published FPGA-based systems, the depth estimation and view synthesis steps are usually treated separately. There are relatively few real-time systems which have combined both parts on FPGAs, such as [10] and [11]. In [10], an FPGA-based system is capable of synthesizing one view from 1080p stereoscopic inputs at 60 fps. A free viewpoint synthesis system utilizing three-camera disparity estimation is presented in [11]. It is implemented on a Virtex 7 FPGA board, achieving a speed of 55 fps for 1024 $\times$ 768 XGA images.

FPGA-based hardware accelerators for depth estimation have been developed in [12]–[16]. The design in [12] outputs 1024 $\times$ 768 pixel images at 60 fps, whereas the designs in [13] and [14] also reach the XGA resolution at 129 fps and 199 fps respectively. A semi-global step is applied in both [15] and [16] to improve the depth accuracy. The hardware system in [15] achieves a throughput of 1600 $\times$ 1200 pixel images at 42 fps, while the design in [16] shows a very promising performance of up to 1280 $\times$ 960 resolution at 197 fps.

DIBR-based view synthesis engines on FPGAs are developed in [17]–[20]. Single-view synthesis engines with a throughput of 1080p images at 60 fps are presented in [17] and [18]. Multi-view rendering solutions for 1080p at 30 fps and 96 fps are developed in [19] and [20], respectively.

After exploring state-of-the-art hardware systems extensively, a complete DIBR system is developed on an FPGA to achieve high synthesis quality and real-time processing speed for high-definition images. A more detailed comparison is given in Section V.

III. Algorithmic Flow

This section is a summary of the proposed DIBR algorithmic flow. Further, we explain the specific selection and

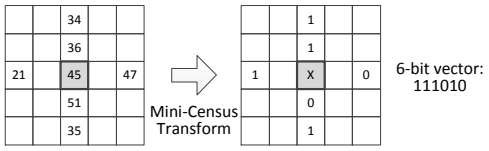


Fig. 3. The mini-census transform.

parametrization of the methods involved with respect to hardware efficiency. Using a stereo image pair, disparity maps can be produced with a local disparity estimation method that focuses on depth discontinuities and disparity smoothness to improve the synthesis quality. Then virtual views can be rendered by view synthesis that consists of view interpolation and extrapolation. The individual steps are explained in more details below.

A. Disparity Estimation

Disparity estimation algorithms can be divided into two groups: local approaches and global approaches [21]. Since local approaches only utilize local information, the accuracy is usually not sufficient in textureless and occluded regions. On the other hand, while global approaches can show better results, they are not easily implemented using dedicated hardware due to their high computation complexity and irregular data access [22]. Although several hardware systems based on global approaches have been recently presented [15] [16], the majority of existing hardware implementations use local approaches.

For the purpose of generating high-quality views, two important factors must be present in disparity maps [23]. First, depth discontinuities should be correctly estimated and placed at the right location. Second, the maps should be smooth in areas of constant depth to avoid artifacts in virtual views. Considering the two factors, a local disparity estimation method is proposed with regard to its applicability in hardware setups. The computational flow of disparity estimation can be summarized in four well-defined steps: cost calculation, cost aggregation, disparity selection and disparity refinement [21]. The details of each step are explained as follows.

1) Cost Calculation: The census transform is a commonly used function to calculate the matching cost between the chosen stereo regions. The mini-census transform, a sparse version of census transform, is first proposed in [24]. It has been proved that the mini-census transform which intends to reduce hardware resource requirements is able to deliver better results than the full census transform [25].

The disparity estimation method begins by applying the mini-census transform. It converts 8-bit luminance of every pixel into a 6-bit vector where each bit corresponds to a certain pixel in the local neighborhood around the pixel of interest. As shown in Fig. 3, a bit will be set if the corresponding pixel has a lower luminance than the center pixel which is marked by gray. Consequently, the pixel characteristic is no longer represented by its luminance, but rather by the 6-bit vector that includes the relative luminance difference. This transformation reduces the on-chip storage and makes disparity estimation robust to radiometric changes [22]. Once the input images have been transformed, the matching cost is defined as the Hamming distance between the vectors.

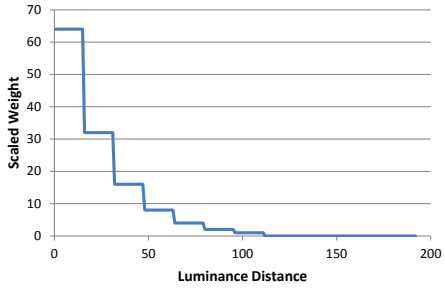


Fig. 4. The curve of the approximated weight function.

2) Cost Aggregation: This is the most important step in local estimation methods because it largely influences the disparity accuracy and computation complexity. In order to obtain accurate discontinuities in disparity maps, a segmentation-based adaptive support weight (ADSW) method [26] is used here, which assumes that segment boundaries in an image are also depth discontinuity boundaries. The function of the weight coefficient w is defined as

$$w(p_i, p_c) = \begin{cases} 1.0 & p_i \in S_c \\ \exp(-\frac{d_c(I(p_i), I(p_c))}{\gamma_c}) & \text{otherwise} \end{cases} \quad (1)$$

where p_i is any pixel in the support region, S_c is the segment of the central pixel p_c, d_c is the Euclidean distance in the CIELAB color space, and γ_c is a fixed parameter [26]. It is assumed that pixels inside the segment S_c have a similar disparity value and their weights are equal to the maximum value of the range; while pixels outside S_c usually have different disparity values and their weights are calculated based on their color distances to p_c. Finally, the aggregated cost is calculated by summing up all the weighted matching costs in the support region, and then normalized with the sum of weight coefficients.

To reduce the computation complexity and make the segmentation-based ADSW method more hardware-friendly, some optimizations are proposed. (1) The input images are divided into segments using thresholding instead of the mean shift method used in [26]. (2) The YUV color representation is adopted instead of the CIELAB color representation to replace signed floating-point numbers with unsigned integers. In our design, only the luminance (Y) channel from the YUV representation is used to reduce the potential bandwidth and storage requirements. (3) Rather than Euclidean distance, Manhattan distance is used to avoid square and square root computations. (4) The $exp(-x)$ function is approximated by the $2^{(6-x)}$ function, and the curve of the weight coefficient is shown in Fig. 4. It will assign a maximum weight of 64 if the luminance distance is less than 16 and a minimum weight of 0 if the luminance distance is greater than 128. As a result, the multiplication between the initial matching costs and the weight coefficients is reduced to a left shift operation.

Table I lists how the accuracy of disparity maps is impacted by these optimization techniques. The error rate is averaged

TABLE I
IMPACT OF THE OPTIMIZATIONS ON ERROR RATE

Optimizations	1	2	3	4	Overall
Error rate	+ 0.4%	+ 2.4%	- 1.4%	+ 2.5%	+ 3.9%

1 - Thresholding segmentation. 2 - Y color representation. 3 - Manhattan distance. 4 - Approximated $2^{(6-x)}$ function.

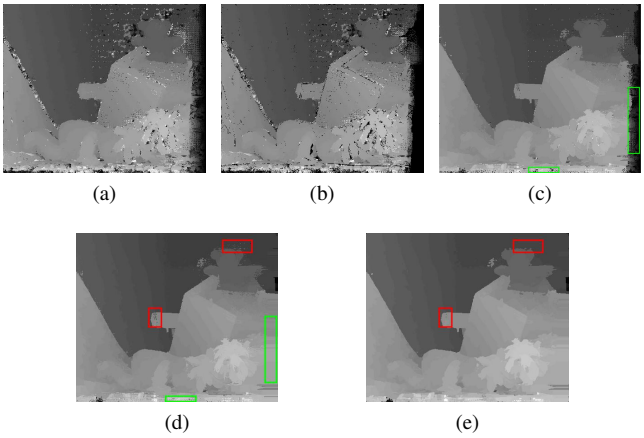


Fig. 5. The process of disparity refinement on the right scene of *Teddy*. (a) The initial disparity map. (b) The disparity map after consistency check. (c) The disparity map after disparity voting. (d) The disparity map after disparity inpainting. (e) The final disparity map after median filtering.

over the *Tsukuba*, *Venus*, *Teddy* and *Cones* stereo images from the Middlebury benchmark. The overall error rate is increased by 3.9% after the integration of all optimization techniques.

3) Disparity Selection: The aforementioned cost aggregation step is executed for the whole disparity range and the disparity with the minimum aggregated matching cost is attainable through a winner-takes-all (WTA) method. In fact, the WTA is widely used in most of the local disparity estimation methods [21].

4) Disparity Refinement: As shown in Fig. 5a, initial disparity maps are obtained including occlusions around object edges and at the image borders. To improve the accuracy of the initial disparity maps, a segmentation-based refinement which consists of consistency check, disparity voting, disparity inpainting and median filtering is proposed.

A segmentation-based consistency check is proposed to determine whether disparity maps are valid or not. The right-to-left check means that for each pixel p of its right disparity $D_r(p)$, the corresponding pixel p' is determined in the left image based on $D_r(p)$, then the disparity $D_l(p')$ and the segment $S_l(p')$ of the left image are compared with $D_r(p)$ and $S_r(p)$ of the right image respectively. If both values are equal, the right disparity $D_r(p)$ will be marked as valid. This is expressed as

$$D_r(p) = \begin{cases} valid & \text{if } D_r(p) = D_l(p') \ \& \ S_r(p) = S_l(p') \\ invalid & \text{otherwise} \end{cases} \tag{2}$$

with p now related to p' as $x_{p'} = x_p + D_r(p)$, $y_{p'} = y_p$. The checked right disparity map is shown in Fig. 5b, and pure black patches are disparities that have been invalidated. The process is then reversed for the left-to-right check of the left disparity map D_l.

TABLE II
IMPACT OF THE REFINEMENT STEPS ON ERROR RATE

	Venus	*Teddy*	*Cones*	Average
Disparity voting	- 68.3%	- 30.3%	- 25.6%	- 41.4%
Disparity inpainting	- 9.9%	- 15.1%	- 14.2%	- 13.1%
Median filtering	- 2.62%	- 2.78%	- 5.56%	- 3.65%
Overall	- 80.8%	- 48.2%	- 45.4%	- 58.1%

After the consistency check, the disparity voting updates the center disparity with the most frequent valid disparity inside its local support window. It works because valid pixels in the same window have a high probability of belonging to the same object and should have the same disparity. Disparity voting is able to reliably fill in the occlusions and smooth out disparities over patches, as shown in Fig. 5c.

Although the disparity voting removes many invalid disparities, it will fail if the window does not contain any valid disparities. This occurs mostly near the borders in Fig. 5c, and can also manifest itself where the occlusions are too large. In order to address the problem, disparity inpainting is used to replace every invalid disparity with the closest valid disparity on its scanline. The visual improvements are shown in the green boxes in Fig. 5c and Fig. 5d.

Finally, small disparity outliers, such as speckle noise, are filtered using a median filter. The effect of the median filtering is indicated by the red boxes in Fig. 5d and Fig. 5e.

As shown in Fig. 5, the proposed segmentation-based refinement contributes significantly to smooth disparity maps with sharp object edges and little to no artifacts. Furthermore, a quantitative measurement on *Venus*, *Teddy* and *Cones* is summarized in Table II. Note that *Tsukuba* is not measured here because its image borders are neglected in the ground truth. The average error rate is reduced by 58.1% after the refinement, where the disparity voting, disparity inpainting and median filtering contribute 41.4%, 13.1% and 3.65% to the reduction, respectively.

B. View Synthesis

The final generated disparity maps can be used to render virtual views. Since the input images have been rectified, the rendering simplifies to a horizontal pixel shift depending on the pixel's disparity value.

According to the target location of the virtual views, view synthesis can be classified into view interpolation and view extrapolation. For the locations on the baseline between the left and right reference views, view interpolation is used with both images and their disparity maps; for other locations on the baseline, view extrapolation is processed using only one image and the corresponding disparity map.

In both cases, the rendering is done with a two-step warping method. The method first warps disparity maps to the target location, and then uses them to generate virtual images. It has been proved in [27] that the two-step warping could perform better than the one-step warping which directly warps the

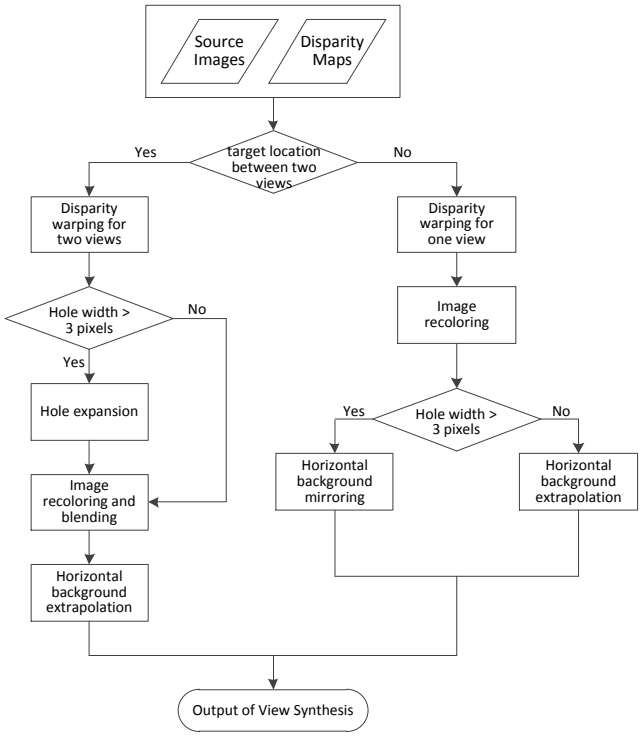


Fig. 6. The flowchart of view synthesis.

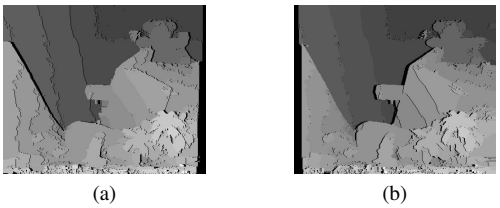


Fig. 7. The disparity maps are warped to the target location. (a) From the left view. (b) From the right view.

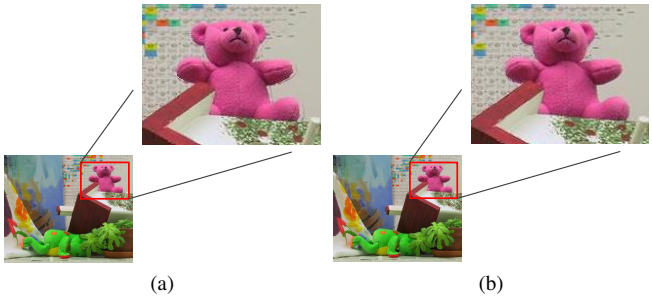


Fig. 8. The effect of hole expansion is demonstrated in the magnified part of the virtual images. (a) Without hole expansion. (b) With hole expansion.

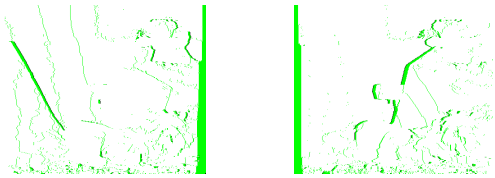
images, because the sampling precision is higher in the virtual views. Note that the warping is not a one-to-one mapping, and there may be multiple pixels warped to the same pixel location. Thus, not all pixels in the virtual images will be filled in and the images will still contain some blanking holes. They are caused not only by occlusions of the source images, but also by mismatches and noise in the disparity maps.

To handle the holes, different methods are proposed for view interpolation and extrapolation, respectively. In the view interpolation, both reference views are warped to the target location independently, and then the two virtual views are used to complement each other because most of the occluded regions could be seen by either one of the two views. Besides the complementary process, hole expansion is proposed for large holes to avoid visual disturbances on the border of the areas where only one view is available. In addition, horizontal background extrapolation [28] is used to fill in the holes that can not be found from both views. In the view extrapolation, the target location is not between the left and right reference views, and only the nearer view is used to render the virtual view. A dynamically adaptive inpainting method, which contains horizontal background extrapolation and mirroring, is utilized to fill in the holes depending on their width. The flowchart of the proposed view synthesis algorithm is illustrated in Fig. 6. On the basis of our experiments, 3 pixels width is adopted as the threshold value to determine a hole whether it is large or not. Algorithm details and experimental comparisons are explained below.

1) View Interpolation: Both disparity maps of the left and right reference views are first warped to the target location independently, as shown in Fig. 7. The warping of each pixel is determined by its disparity. When multiple pixels are warped to the same pixel location, the pixel with the greatest disparity value will be chosen because foreground objects that have large disparities may occlude background objects with small disparities under the parallel stereo camera configuration. After the disparity warping, there are some blanking holes to which none of the pixels have been warped, indicated as the pure black patches in Fig. 7.

Given the warped disparity maps, the colors are fetched from the source images to produce two virtual images. For pixels that can only be seen in either one of the virtual images, they are recolored directly to fill in most of the holes; for pixels that can be seen in both virtual images, they are recolored and then blended to generate average color values. This blending process improves the color consistency between the two source images with higher quality. However, a contour of visual disturbances surrounding the foreground objects is noticeable in the final virtual image, as shown in Fig. 8a. The halo-like effect is caused by color differences on the border of the areas where only one view is available. To address it, hole expansion is proposed for large holes in the warped disparity maps. It expands the hole regions towards the warping direction by one pixel so that more background information is copied from the complementary view. The hole expansion process is shown in Fig. 9. As a result, the colors are prevented from suddenly changing at the hole borders, and visual improvements could be seen in Fig. 8b.

It is noted that there are still some remaining holes that can not be found from both reference views. This problem can be solved by inpainting methods. Here, a simple method called horizontal background extrapolation is employed, which fills in holes by horizontally extrapolating the pixels on the border of background objects [28]. The disparity information is taken into account to avoid using foreground objects that are visible



(a) (b)

Fig. 9. Hole expansion in the warped disparity maps. Green pixels represent the initial hole regions, and black pixels represent the hole expansion regions. (a) From the left view. (b) From the right view.

(a) (b)

Fig. 10. The virtual images using different inpainting methods. (a) Horizontal background extrapolation only. (b) The dynamically adaptive inpainting method which contains horizontal background extrapolation and mirroring.

and not occluded. Each pixel inside the hole H is filled by horizontal background extrapolation according to:

$$I(x,y) = \begin{cases} I(m_l, y) & \text{if } d(m_l, y) < d(m_r, y) \\ I(m_r, y) & \text{otherwise} \end{cases} \tag{3}$$

where (m_l, y) and (m_r, y) are the left and right positions of the first boundary pixels of the hole H in the scan line of pixel (x, y), which has to be filled, and d denotes the disparity values of those pixels. As pointed out in [28], the horizontal background extrapolation has satisfactory visual effect without taking too much computation power.

2) View Extrapolation: When the target location is not between the left and right reference views, only the disparity map of the nearer view will be warped. Then the virtual view is recolored from the corresponding source image using the warped disparity map. In this case, a number of large holes will appear in the virtual images due to the lack of the complementary process. They also need to be filled by inpainting methods.

Although the horizontal background extrapolation is an effective inpainting method, it might cause some undesirable visual artifacts when filling in large holes. Thus, a horizontal background mirroring method [29] is used to handle large holes here. As opposed to extrapolation, the hole H is now filled by completely mirroring background pixels, but not only using the first boundary pixels:

$$I(x,y) = \begin{cases} I(m_l - (x - m_l), y) & \text{if } d(m_l, y) < d(m_r, y) \\ I(m_r + (m_r - x), y) & \text{otherwise} \end{cases}. \tag{4}$$

The pixels inside the holes are symmetric with respect to the border of background objects.

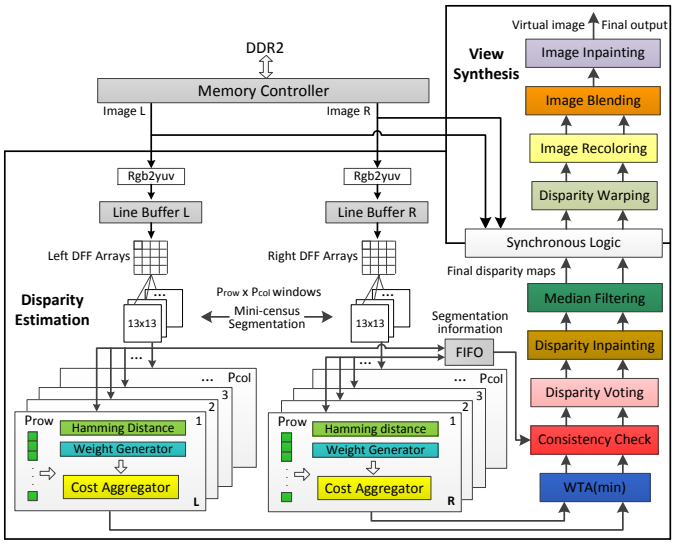


Fig. 11. Top-level block diagram of the proposed hardware system.

In the view extrapolation, horizontal background extrapolation and mirroring are dynamically used depending on the width of the holes. The final virtual images using different inpainting methods are displayed in Fig. 10. It is observed that the dynamically adaptive inpainting method performs much better in the yellow boxes.

IV. Hardware Implementation

A. System Overview

The hardware system of the proposed DIBR algorithm is shown in Fig. 11, which consists of two main modules: disparity estimation and view synthesis. The original RGB data is first read back from the DDR2 memory sequentially and stored into line buffers. Then the disparity maps of the stereo image pair are calculated in the disparity estimation module with a $P_{row} \times P_{col}$ parallelism scheme. Finally, the stereo images and the corresponding disparity maps are synchronized and buffered as the input of the view synthesis module, where the virtual image is generated as the final result.

For the purpose of achieving real-time processing speed at high-definition images, the system is designed to be a high throughput architecture. However, external memory bandwidth is always an important limitation. For example, it requires 1.24 GB/s for 1080p at 50 fps considering loading and reading each image pair one time. The DDR2 memory that is mounted on FPGA prototyping boards can typically reach approximately 5 GB/s. Under this condition, hardware implementations that require multiple reads of a pixel can easily exceed the bandwidth limitation. Using multiple cameras in one system may also pose external memory bandwidth challenges. The hardware in [24] needs to access external memory at least five times for every pixel, and thus it requires high memory bandwidth even for low resolution images. In the proposed hardware system, the data allocation scheme requires reading each pixel only once from the external memory during the whole process.

The high-speed processing is also guaranteed by the parallelism and pipeline scheme in the system. As shown in

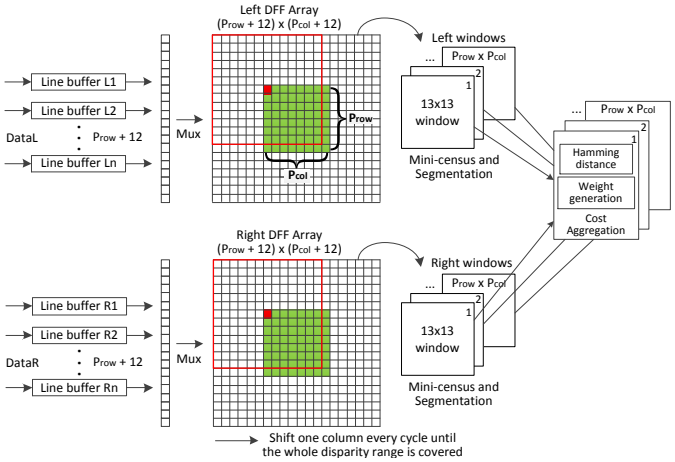


Fig. 12. The parallelism scheme in the disparity estimation module.

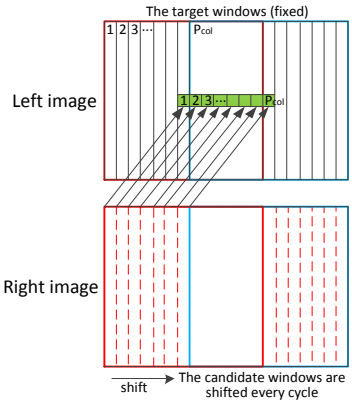


Fig. 13. The proposed PWR technique.

Fig. 11, the stereo images are processed in parallel whenever possible. However, it is almost impossible to connect the two modules directly because they have different requirements in accessing the memory. The disparity estimation module reads the data repeatedly to cover the disparity range of different pixels, and then writes disparity maps out in scanline order; the view synthesis module requires image data and disparity maps synchronously, and the data from the two images has to be accessed in reverse order for each other. In such a setting, ping-pong line buffers are used to support fully row-based pipelined processing. Further, the implementation details inside each module will be discussed in the following subsections.

B. Disparity Estimation Module

1) Parallelism Scheme: During the disparity estimation, it is challenging to develop an efficient parallelism scheme due to the requirement for real-time processing speed. Many hardware systems calculate the disparity maps pixel by pixel using progressive scan [12] [24]. It is simple to implement but inefficient. In our system, a parallelism scheme which combines the row-level parallelism P_{row} with the column-level parallelism P_{col} is proposed. It means that P_{row} pixels in neighboring rows and P_{col} pixels in neighboring columns are processed in parallel. Thus, the parallelism degree is $P_{row} \times P_{col}$ in the system.

The estimation process starts with converting the RGB pixels into the YUV format. As shown in Fig. 12, the 8-bit luminance (Y) components of the pixels are loaded into the D flip-flop (DFF) Arrays, the size of which is $(P_{row} + 12) \times (P_{col} + 12)$ depending on the parallelism degree. The colored $P_{row} \times P_{col}$ pixels are processed in parallel, and the support region of each pixel is selected from the DFF Arrays as a 13 $\times$ 13 window block, which is indicated by the red box for the red pixel. In the $P_{row} \times P_{col}$ window blocks, the mini-census transform and the thresholding segmentation are operated for the Hamming distances and weight coefficients, respectively.

To calculate the left disparity map in Fig. 12, the left pixels are kept unchanged in the left DFF Array during the search process, while the right pixels are read as candidate pixels continuously. They are loaded into the right DFF Array and shifted one column every cycle until the whole disparity range (D_r) is covered. The right disparity map is calculated in the same way, and the left pixels are shifted by column as candidate pixels. Thus, the disparity estimation module is able to process $P_{row} \times P_{col}$ pixels of each image every $(13 + D_r)$ clock cycles. Here the disparity range can be configured by the user depending on the expected distance to the objects. And it is noted that configuring the module for a low disparity range can increase the processing speed. The relationship between the processing speed and parameter settings (e.g. P_{row}, P_{col} and D_r) will be discussed in detail in Section V-B.

2) Data Reuse: To reduce the memory bandwidth and computation requirements, data reuse techniques are used in the disparity estimation module. An efficient method, partial column reuse (PCR), has been proposed in [24]. It reuses the data in each column, which is usually a part of multiple horizontally overlapped window blocks. Therefore, the data of each column can be shared by the calculation in these windows. In our system, the PCR is adopted not only for columns horizontally, but also for rows vertically.

Further, a parallel window reuse (PWR) technique is proposed based on our parallelism scheme. The PWR reuses the window blocks for the P_{col} neighboring pixels in the same row. To calculate the disparities of the left image, the pixels from the right image are shifted as candidate pixels every cycle, and the right window blocks can be used P_{col} times from pixel 1 to pixel P_{col}. As shown in Fig. 13, the window block in the red solid lines is first processed for pixel 1. Then in the next cycle, its location is shifted to right by one column and the block is processed for pixel 2, as indicated by the red dotted lines. This operation continues until the window block is used for pixel P_{col}, and its location is shown in the blue solid lines. As a result, the mini-census transform and the thresholding segmentation, which are operated in the window blocks, can be calculated only once for the P_{col} pixels. In every cycle of the search process, only the window blocks that contain new shifted columns are calculated and repetitive computations are avoided.

3) Cost Aggregation: In each aggregation submodule, the Hamming distances and weight coefficients are produced using the left and right window blocks. The Hamming distances are calculated as matching costs between the mini-census

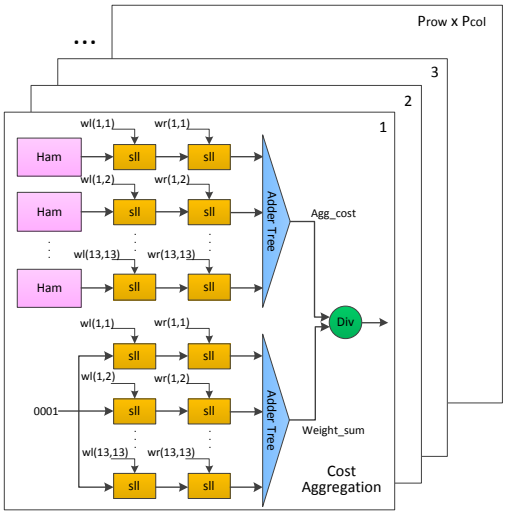


Fig. 14. The architecture of the aggregation submodule.

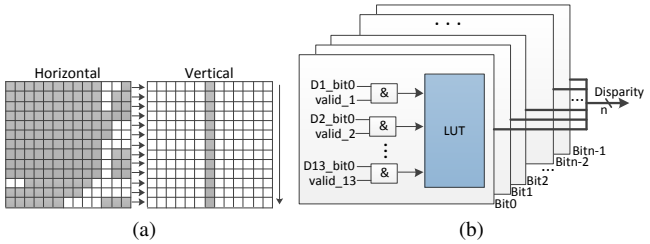


Fig. 15. Two techniques used in the disparity voting. (a) The horizontal-vertical approach. (b) The architecture of the bitwise fast voting.

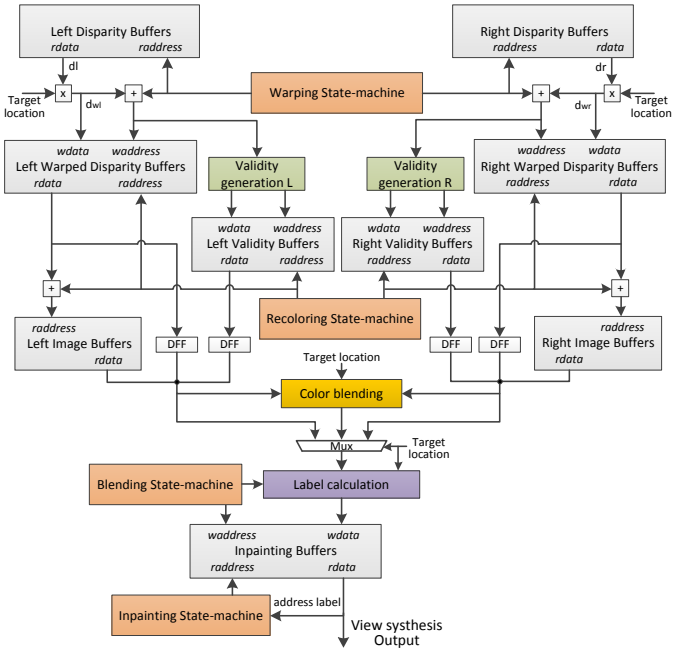


Fig. 16. The architecture of the view synthesis module.

data from both windows. With the help of the segmentation information, the weight coefficients are generated using a look-up table (LUT). It is a straightforward solution that consumes a low amount of hardware resources. The matching costs are multiplied by the corresponding weight coefficients, and this multiplication is reduced to a left shift operation due to the approximated weight function. The final aggregated cost is calculated by summing the weighted costs with a tree adder, then dividing it by the sum of weight coefficients. The architecture of the aggregation submodule is illustrated in Fig. 14. A total of $P_{row} \times P_{col}$ aggregation submodules are processed to deal with the $P_{row} \times P_{col}$ parallelism degree.

After the cost aggregation, the disparity with the minimum aggregated cost is selected by the WTA method. In addition, the segmentation information of each pixel is stored in a FIFO for the segmentation-based refinement.

4) Disparity Refinement: Consistency check, disparity voting, disparity inpainting and median filtering are implemented in a pipeline to generate the final disparity maps. First, the initial disparity maps of both images are used to check the consistency of every pixel with the help of the segmentation information. One more bit is generated to label whether each disparity is valid or not.

Then the disparity voting is processed in a 13 $\times$ 13 support window to achieve a reliable result. Two techniques, which have been proposed in [30], are applied here for fast and efficient implementation. One technique is that the voting can be modified to a horizontal-vertical approach, as shown in Fig. 15a. The gray blocks represent valid disparities, while the white blocks represent invalid ones. The 2D voting procedure is approximated to two successive 1D voting procedures, and the computation complexity can be reduced from $O(N^2)$ to $O(2N)$. The other technique is using a bitwise fast voting method to handle the most frequent valid disparity. It drives each bit of the disparity value independently from the other bits. In this way, the hardware cost depends on the number of the disparity bits in binary. The architecture of the bitwise fast voting for one row is shown in Fig. 15b. It is noted that when counting bit votes, the valid information of each disparity must be taken into account. Since the support window size is 13 $\times$ 13, the 13 most frequent disparities in 13 rows are first derived. Then the most frequent disparity in the support window is selected from the 13 derived ones as the center disparity.

After the disparity voting, most invalid disparities are updated to valid ones only by their valid neighbours. The remaining invalid disparities are replaced with the closest valid ones for the disparity inpainting. Finally, the median filtering is processed, and its hardware architecture proposed in [31] is implemented in our system. The refined disparity maps are written into the disparity buffers in scanline order.

C. View Synthesis Module

In the view synthesis module, the image data and disparity maps have been synchronized in buffers. The target location is defined as an unsigned integer. As shown in Fig. 16, this module is split into 4 concurrent state machines: *Warping*, *Recoloring*, *Blending* and *Inpainting*; and ping-pong buffers are used between the states to provide pipelined processing.

The *Warping* state will start once the disparity and image buffers are available. The read address is generated to read

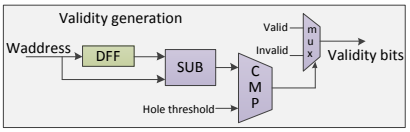


Fig. 17. The architecture of the validity generation submodule.

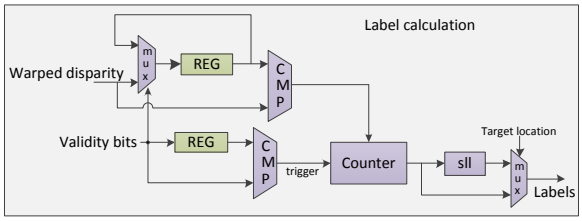


Fig. 18. The architecture of the label calculation submodule.

the disparity values d_l and d_r from the disparity buffers. Then the warped disparity values d_{wl} and d_{wr} are calculated using d_l and d_r; they are also used to generate the write address to the warped disparity buffers. In addition, the corresponding validity bits of d_{wl} and d_{wr} are produced in the validity generation, where the hole expansion is implemented depending on the hole width. The architecture of the validity generation submodule is shown in Fig. 17. If a disparity is not validated, it will be considered in the holes. Whenever d_{wl} and d_{wr} are written into the warped disparity buffers, their corresponding validity bits are written to the same address of the validity buffers.

The *Recoloring* state works when the warped disparity and validity buffers are available. The d_{wl} and d_{wr} are used to calculate the read address to the image buffers. For the view interpolation, the read data from both images is processed in the color blending submodule; for the view extrapolation, only the buffers of one image are read according to the target location, and the read data is bypassed without the blending.

In the *Blending* state, the warped disparity values and validity bits are processed in the label calculation submodule. As shown in Fig. 18, a label is generated for each pixel inside the holes to indicate the address of its corresponding inpainting pixel. Then the image data and address labels are written into the inpainting buffers.

Finally in the *Inpainting* state, the read address is generated using the address labels to fill in the holes, and the image data is read back from the inpainting buffers. The virtual image is sent out pixel by pixel in scanline order.

V. Experimental Results

A prototype of the proposed system has been implemented on an Altera EP4SGX530 FPGA board. It is evaluated using rectified synthetic stereo images, which are initially stored in the DDR2 memory. Two important aspects — processing speed and accuracy evaluation — are elaborated in the following subsections to prove the effectiveness of the proposed DIBR system. In addition, to make the system more flexible, its scalability is also discussed below.

A. Overall Hardware Performance

An overall comparison is first made among our system and other complete DIBR systems which are listed in the related work. Further, there are relatively few published FPGA-based systems that implement the disparity estimation and view synthesis units together. Both parts are usually treated as separate subproblems in the literature. Therefore, the two subsystems are then compared with related hardware implementations on FPGAs.

1) Complete Systems: Complete DIBR systems with similar functionality are presented in [6], [7] and [9]–[11], which are listed at the top of Table III. The systems in [6] and [9] are based on CPU and GPU respectively, but it is noted that they can not achieve real-time processing speed for high resolutions. The work in [7] is implemented on a dual processor workstation with two Intel Xeon5690 CPUs and two NVIDIA GTX590 graphics cards. It synthesizes eight interleaved views from 1080p input images at 24 fps. Clearly, the size of this system, the cost, and power consumption make it unsuitable for integration into consumer devices. In [10], the synthesis system is implemented on a Stratix III FPGA, which is able to produce one virtual view generated from 1080p content at 60 fps. Since no detailed FPGA results are published for the system, its hardware complexity is provided in 250 nm CMOS technology. Compared to our system, only the view interpolation is implemented in [10], where the virtual viewpoints are limited between the stereoscopic input images. The system in [11] utilizes XGA images from three cameras and achieves a speed of 55 fps. Therefore, it is able to reach only 21 fps when its performance is scaled to 1080p images. This FPGA-based system is implemented on a Virtex 7 board, and occupies quite a large number of hardware resources. As a comparison, our proposed system, which can run up to 120 MHz on an Altera EP4SGX530 FPGA, occupies fewer resources comprising 254k LUTs, 149k registers and 3.73M RAM bits, and supports 1080p output resolution at 45 fps.

2) Disparity Estimation Cores: In DIBR frameworks, disparity estimators such as the implementations in [12]–[16] can be viewed as an individual step. A quantitative comparison of these FPGA-based estimators is shown in Table III in the middle, which focuses on three aspects: disparity accuracy, processing speed and resource utilization. The disparity accuracy is measured by the error rate and will be discussed in detail in Section V-C.

The processing speed is evaluated by million disparity estimations per second (MDE/s), which means the product of frame rate, image resolution and disparity range. In the proposed system, the disparity estimation core is able to reach 45 fps for 1080p images with a disparity range of 64 pixels. Among the listed estimators, it is the only one designed for 1080p resolution, and can achieve 11944 MDE/s on an Altera EP4SGX530 FPGA. Some estimators in [13], [15] and [16] achieve a similar or even better processing speed. But the estimator in [13] uses a simpler local algorithm based on variable support region, which will cause a worse accuracy performance than that of our disparity estimation core. Semi-global methods are applied in [15] and [16] to achieve a

TABLE III
OVERALL HARDWARE COMPARISON WITH OTHER COMPLETE DIBR SYSTEMS, DISPARITY ESTIMATION, AND VIEW SYNTHESIS UNITS

Complete systems	Input	Output	Technology	LUT	Reg	Ram [MBit]	f [MHz]	FPS
This work	2×1080p	1×1080p	Stratix IV	254k	149k	3.73	120	45
Xu et al. [6]	2×VGA	1×VGA	CPU	Intel Core2 6700 CPU			-	8
Riechert et al. [7]	2×1080p	8mix 1080p	GPU+CPU	2×Xeon5690, 2×GTX590			-	24
Dumont [9]	2×SVGA	1×SVGA	GPU	2×GTX TITAN Black			-	10
Liao et al. [10]	2×1080p	1×1080p	ASIC 250nm	470kGE		1.27*	-	60
Akin et al. [11]	3×XGA	1×XGA	Virtex 7	607k	303k	8.43*	175	55
Disparity estimation	MDE/s†	Output (D_r°)	Technology	LUT	Reg	Ram [MBit]	Average error rate	FPS
This work	11944	2×1080p (64)	Stratix IV	247k	144k	3.31	6.02%	45
Zhang et al. [12]	3019	1×XGA (64)	Stratix III	80.6k	94.9k	3.77	8.20%	60
MCADSR [13]	13076	1×XGA (128)	Stratix IV	60.2k	33.3k	2.87	7.65%	129
Jin et al. [14]	9362	1×XGA (60)	Virtex 6	123k		6.08	6.05%	199
Wang et al. [15]	10472	1×UXGA (128)	Stratix V	222k	149k	16.6	5.61%	42
Li et al. [16]	15492	1×XVGA (64)	Stratix V	96.1k	83.2k	20.3	6.03%	197
View synthesis	Input	Output	Technology	LUT	Reg	Ram [MBit]	Quality metrics	FPS
This work	2×1080p	1×1080p	Stratix IV	7.61k	4.31k	0.41	PSNR/SSIM	65
Chen et al. [17]	1×360p△	9mix 1080p	Cyclone III	1.40k	1.28k	0.47	Subjective perception	60
Lai et al. [18]	1×1080p	2×1080p	Virtex 5	5.77k	2.43k	-	Subjective perception	60
Jin et al. [19]	2×360p△	9mix 1080p	Virtex 4	5.56k	5.25k	0.29*	Subjective perception	30
Wang et al. [20]	2×1080p	1×1080p	Virtex 6	23.5k	20.5k	3.31*	PSNR	96

* Estimates, not accounting for memory controller and other I/O infrastructure.
† MDE/s: Million disparity estimation per second.
° D_r: Number of the disparity range.
△ 640×360 resolution.

high processing speed, but much more memory resources are needed to store temporary results.

The resource utilization on FPGAs is another important criterion especially when hardware resources are limited. It is an estimated evaluation because the basic units of different FPGA platforms are not the same. It is noted that the proposed disparity estimation core occupies more logic resources than others in Table III. There are two main reasons. One is that the disparity maps of both stereo images are simultaneously generated, which will be used in the view synthesis core; the other is that the proposed algorithm uses many promising methods (e.g. the segmentation-based ADSW and refinement) to achieve a high accuracy performance.

However, it is difficult to provide a unified metric which includes all the mentioned aspects, because the processing speed and disparity accuracy are in quite different domains. Besides, the importance of the aspects varies over the requirements of different applications. Thus, it is more meaningful to discuss the metric in a specific scenario in the future.

3) View Synthesis Cores: A comparison of the proposed view synthesis core with related FPGA-based implementations is given in Table III at the bottom. Interleaved view synthesis engines are developed in [17] and [19] with performances of up to 1080p at 60 fps and 30 fps, respectively. In contrast, architectures that are able to render virtual views directly are designed in [18] and [20]. Among the listed designs, the work in [20] achieves the highest processing speed of 96 fps while it occupies the most hardware resources. The proposed work is able to produce one virtual view of 1080p at 65 fps. Note that it occupies a little more resources than those the designs in [17]–[19] occupy. A main reason for this is that the view interpolation and extrapolation are both implemented in our view synthesis core. Additional reasons are the different image and disparity map resolutions.

The synthesis quality of the virtual views is also an important criterion to these designs. The ultimate validation is whether the views look real or not, but sometimes it is difficult to be measured only depending on the subjective human perception. For the proposed view synthesis core, quantitative metrics such as PSNR and SSIM [32] are used to evaluate the synthesis quality. More details will be discussed in Section V-C.

B. Scalability

A key feature of the proposed system is scalability. As shown in Table III, the disparity estimation core occupies most of the hardware resources in the whole system. Thus, the disparity estimation core can be scaled up to make the system more flexible, and a tradeoff is made between the processing speed and resource utilization by tuning the parameters such as the image resolution $H \times W$, disparity range D_r, and parallelism degree $P_{row} \times P_{col}$. The processing speed (frame per second) and resource utilization of the disparity estimation core are shown in Table IV with different parameters. The values are based on the compilation results of each module.

A further analysis is given to the relationship between the processing speed and parameter settings. The frame rate of the disparity estimation core could be estimated as:

$$FPS \approx \frac{f \times P_{row} \times P_{col}}{H \times W \times (D_r + 13)} \tag{5}$$

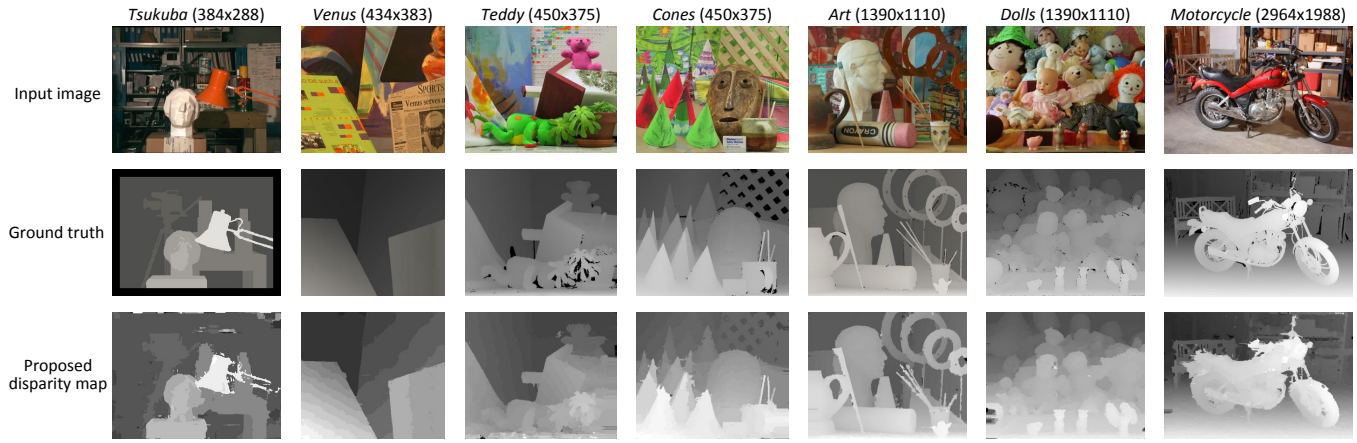


Fig. 19. True disparity maps and experimental results of the disparity estimation core.

TABLE IV
PROCESSING SPEED AND RESOURCE UTILIZATION OF THE DISPARITY ESTIMATION CORE WITH DIFFERENT PARAMETERS

Image resolution	Parallelism		FPS	Resource utilization		
	P_{row}	P_{col}		LUT	Reg	RAM Bits
VGA	4	1	24.59	43246	20686	938112
D_r=48	4	2	49.17	59714	27818	939072
	8	2	99.46	83935	45152	1143872
XGA	4	2	15.26	59623	28942	1465348
D_r=64	4	4	30.52	87048	43488	1467456
	8	4	61.37	138428	76886	1795136
1080p	4	4	11.18	88263	45084	2700312
D_r=64	8	4	22.37	139897	78478	3311136
	8	8	45.31	246791	144526	3314752

where f is the frequency of the system and can run up to 120 MHz. It is noted that the frame rate increases with increasing parallelism degree $P_{row} \times P_{col}$, but decreases with larger image resolution $H \times W$ and disparity range D_r.

The total resource utilization of the disparity estimation core is also estimated here:

$$LUT \approx 3150 \times P_{row} \times P_{col} + 1660 \times P_{col} + 31000 \quad (6)$$

$$Reg \approx 2050 \times P_{row} \times P_{col} + 12500 \quad (7)$$

$$RAM \approx 16 \times W \times (66 + 5 \times P_{row}) + 58500. \quad (8)$$

When the parallelism degree $P_{row} \times P_{col}$ is fixed, the logic resource utilization mainly depends on P_{col}, while the memory resource utilization mainly depends on P_{row}.

Due to unpredictable optimization in compilation tools, the real processing speed and resource utilization of the whole core may be a little different from the estimated values. Consequently, the parameter settings in the proposed system could be adjusted according to the resource specification of various FPGA boards.

C. Quality Evaluation

To discuss the quality performance, the disparity estimation and view synthesis cores are first evaluated on the Middlebury benchmark using different metrics, respectively. Then the proposed system is integrally evaluated to prove its effectiveness.

1) The Disparity Estimation Core: For disparity maps, the error rate is utilized as a commonly accepted metric, which means the percentage of bad pixels in different regions [21]. The Middlebury benchmark is widely used in evaluating the accuracy of disparity estimation algorithms. The disparity maps of four image pairs *Tsukuba, Venus, Teddy* and *Cones* are shown in Fig. 19, and quantitative evaluation results are listed in Table V, which also provides a comparison with some state-of-the-art FPGA-based disparity estimators. The average error rate is 6.02% in our design. It is observed that the work in [15] shows an average error rate of 5.61%, which ranks first among the implementations. However, as mentioned in Table III, it occupies more memory resources and achieves a lower processing speed than our design. The designs in [14] and [16] show comparable accuracy performance 6.05% and 6.03% respectively, but neither of them is developed for 1080p resolution. To summarize, the comparison shows that the accuracy of the disparity maps in our design is among the best of FPGA-based implementations.

The resolutions of the image pairs *Tsukuba, Venus, Teddy* and *Cones* are all lower than VGA resolution. However, the proposed design can process images at high resolutions. To evaluate the tolerance for resolution variation, some high-definition images in the Middlebury benchmark are utilized. The results of *Art*, *Dolls* and *Motorcycle* are also shown in Fig. 19, and it is noted that clear and smooth disparity maps are provided. The overall error rates are 12.57%, 8.93% and 30.76%, respectively.

2) The View Synthesis Core: The quality performance of the proposed view synthesis core is evaluated by four image sets from the Middlebury benchmark, including *Teddy, Cones, Art* and *Dolls*. For each set, seven images are placed in the same distance between each other and numbered from 1 to 7 (*IM*1-*IM*7). *IM*2, *IM*6 and their corresponding ground truth maps (*GT*2 and *GT*6) are used to synthesize the virtual views. View interpolation is processed at exactly the viewpoints *IM*3, *IM*4 and *IM*5, while view extrapolation is done at the viewpoints *IM*1 and *IM*7. The virtual and real images are then evaluated in terms of PSNR and SSIM, as shown in Table VI.

TABLE V
ACCURACY COMPARISON OF DISPARITY MAPS

Image set	*Tsukuba*			*Venus*			*Teddy*			*Cones*			Average
Evaluation	nonocc[1]	all[2]	disc[3]	nonocc	all	disc	nonocc	all	disc	nonocc	all	disc	Error Rate
Wang et al. [15]	2.39	3.27	8.87	0.38	0.89	1.92	6.08	12.1	15.4	2.12	7.74	6.19	5.61
This work	2.69	3.47	10.8	0.41	0.52	5.07	4.24	8.69	15.6	2.86	7.96	9.97	6.02
Li et al. [16]	2.79	-	-	0.68	-	-	4.18	-	-	2.67	-	-	6.03
Jin et al. [14]	1.66	2.17	7.64	0.40	0.60	1.95	6.79	12.4	17.1	3.34	8.97	9.62	6.05
Ttofis et al. [33]	2.38	3.01	9.38	0.40	0.70	3.62	7.23	12.7	17.2	2.87	8.59	8.27	6.36
MCADSR [13]	3.62	4.15	14.0	0.48	0.87	2.79	7.54	14.7	19.4	3.51	11.1	9.64	7.65
Zhang et al. [12]	3.84	4.34	14.2	1.20	1.68	5.62	7.17	12.6	17.4	5.41	11.0	13.9	8.20
Ttofis et al. [34]	4.48	6.04	12.7	6.01	7.47	18.2	21.5	28.1	28.8	17.1	25.9	25.8	16.8
Jin et al. [35]	9.79	11.6	20.3	3.59	5.27	36.8	12.5	21.5	30.6	7.34	17.6	21.0	17.2

[1] Percentage of bad pixels except for occluded regions.
[2] Percentage of bad pixels in all regions.
[3] Percentage of bad pixels only along discontinuous regions.

TABLE VI
PSNR AND SSIM RESULTS OF THE VIRTUAL VIEWS

Metric	Image set	*IM*1	*IM*3	*IM*4	*IM*5	*IM*7	Average
PSNR (dB)	*Teddy*	29.03	31.49	31.60	30.69	29.12	30.39
	Cones	27.06	29.89	29.19	29.42	27.31	28.57
	Art	29.26	31.81	31.84	31.46	29.37	30.75
	Dolls	30.64	33.94	34.01	33.56	30.84	32.60
SSIM	*Teddy*	0.9285	0.9566	0.9546	0.9419	0.9351	0.9433
	Cones	0.9018	0.9361	0.9210	0.9342	0.9111	0.9208
	Art	0.9251	0.9571	0.9604	0.9568	0.9253	0.9450
	Dolls	0.9458	0.9660	0.9737	0.9691	0.9463	0.9602

TABLE VII
QUALITY COMPARISON OF THE PROPOSED VIEW SYNTHESIS METHOD WITH OTHER METHODS

Metric	Image set	Zhang et al. [36]	Yang et al. [37]	Sun et al. [38]	Lee et al. [39]	Chen et al. [40]	Kuo et al. [41]	**This work**
PSNR (dB)	*Teddy*	22.33	30.33	-	25.73	32.32	24.76	30.39
	Cones	20.89	27.66	-	24.26	28.73	33.93	28.57
	Art	-	-	26.37	-	-	-	30.75
	Dolls	22.14	31.39	29.66	27.69	-	-	32.60
SSIM	*Teddy*	0.7467	0.9230	-	0.8678	0.9923	0.9890	0.9433
	Cones	0.6646	0.8834	-	0.8285	0.9653	0.9890	0.9208
	Art	-	-	0.9740	-	-	-	0.9450
	Dolls	0.7329	0.9356	0.9885	0.9109	-	-	0.9602

PSNR is an objective method for judging image quality, but it only calculates the color difference between two images. SSIM calculates changes in structural information, which is introduced to correlate better with the human visual system. Note that for every image set, the PSNR and SSIM results of *IM*3, *IM*4 and *IM*5 are always greater than those of *IM*1 and *IM*7. Because *IM*3, *IM*4 and *IM*5 are synthesized by both *IM*2 and *IM*6 in the view interpolation process, where more pixel information is utilized; while *IM*1 is generated in the view extrapolation process from *IM*2 only and *IM*7 is from *IM*6 only.

Since very few FPGA-based view synthesis engines provide quantitative evaluation results, some state-of-the-art view synthesis methods [36]–[41] which are implemented on other platforms such as CPUs and GPUs are introduced for a quality comparison, as shown in Table VII. There are some empty values in the table because the image sets used in the presented methods [36]–[41] are slightly different. It is noted that the SSIM values of [40] and [41] are greater than that of this work. Because [40] and [41] have complicated inpainting functions to fill in the holes of the virtual images, while the proposed method utilizes a relatively simple inpainting process for its implementation on an FPGA. In this condition, more artifacts such as distortions and ghosts will appear inside the holes using our view synthesis method. High PSNR values could possibly be produced, since the artifacts may occur only in some small parts of the image, whereas they are expected to present a larger error in SSIM values.

3) The Complete System: In the proposed DIBR system, the disparity maps are generated in the disparity estimation

TABLE VIII
THE QUALITY PERFORMANCE OF THE VIRTUAL VIEWS BY USING THE GROUND TRUTH AND THE GENERATED DISPARITY MAPS

Metric	Disparity	*Teddy*	*Cones*	*Art*	*Dolls*	Average
PSNR (dB)	Ground truth	30.39	28.57	30.75	32.60	30.58
	Generated maps	30.03	28.29	29.81	32.16	30.07
SSIM	Ground truth	0.9433	0.9208	0.9450	0.9602	0.9423
	Generated maps	0.9351	0.9144	0.9289	0.9426	0.9303

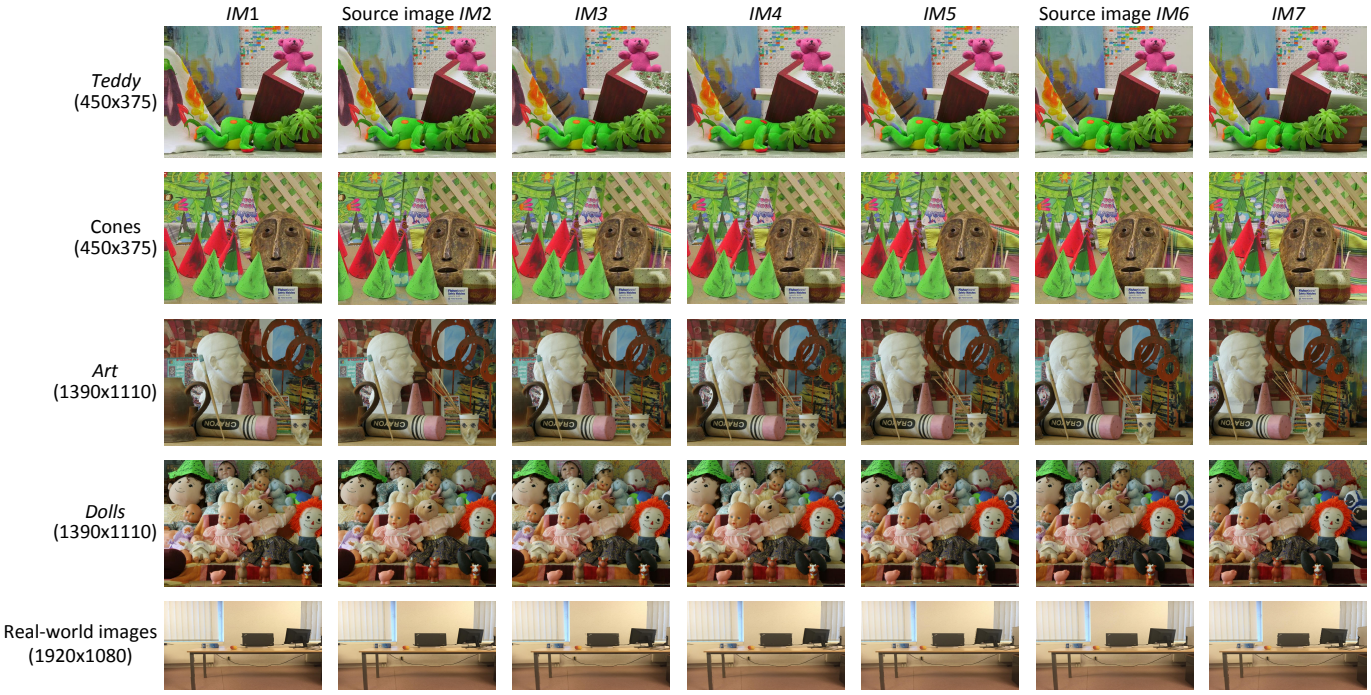


Fig. 20. The virtual images synthesized in the complete system.

core, and then processed in the view synthesis core to produce virtual views. The resulting views are used to evaluate the quality performance of the complete system, and the results are listed in Table VIII, where each number is an average value of *IM*1, *IM*3, *IM*4, *IM*5 and *IM*7. Compared to the evaluation results by using the ground truth, the average PSNR and SSIM values decrease to 30.07 dB and 0.9303, respectively. Because there are bad pixels in the generated disparity maps, while the ground truth is always right.

The subjective perception is also a crucial point for the virtual views. For each image set, two source images (*IM*2 and *IM*6) and five synthesized images (*IM*1, *IM*3, *IM*4, *IM*5 and *IM*7) are shown in Fig. 20. It is observed that no black patches are remaining in the images. The most noticeable artifacts are the left border of *IM*1 and the right border of *IM*7. Since *IM*1 is generated from *IM*2 only and *IM*7 is from *IM*6 only, the pixel information is usually missing at their image borders. As mentioned in Section III-B, the missing content is extrapolated by existing pixels, but it will result in a few repetitive patterns. Besides, some artifacts also appear around object edges. The visual disturbances are caused by incorrect disparity maps in depth discontinuity regions.

The image sets from the Middlebury benchmark are all well captured and rectified so that the results are quite accurate. But the synthesis quality for real-world images may decrease due to some undesirable factors, such as luminance differences and rectification errors. The proposed system is further evaluated by real-world images to prove its robustness, as shown in the last row in Fig. 20. The images are captured in a laboratory at 1080p resolution, and then rectified by software. It is shown that our system still provides high-quality virtual views for the real-world images.

VI. CONCLUSION

This paper presents a complete stereo-view DIBR algorithm which contains disparity estimation and view synthesis. In order to implement the algorithm, a fully parameterized and scalable hardware architecture is designed with hardware-oriented optimizations. Furthermore, a prototype of the DIBR system has been built on an Altera Stratix IV FPGA and can achieve 45 fps for 1080p resolution. The design is evaluated on selected image sets of the Middlebury benchmark; the experimental results have shown that the proposed system has the top-performing processing speed and its accuracy performance is among the best of state-of-the-art hardware implementations. In the future, we will start the application specific integrated circuit (ASIC) design of the DIBR system

for lower cost and power. This could make it more practicable in various 3D applications nowadays.

REFERENCES

[1] N. A. Dodgson, "Autostereoscopic 3D displays," *Computer*, vol. 38, no. 8, pp. 31–36, Aug 2005.

[2] C. Fehn, "Depth-image-based rendering (DIBR), compression, and transmission for a new approach on 3D-TV," in *Proc. SPIE Conf. on Stereoscopic Displays and Virtual Reality Systems*, vol. 5291, 2004, pp. 93–104.

[3] S. F. Tsai, C. C. Cheng, C. T. Li, and L. G. Chen, "A real-time 1080p 2D-to-3D video conversion system," *IEEE Transactions on Consumer Electronics*, vol. 57, pp. 915–922, May 2011.

[4] http://vision.middlebury.edu/stereo/.

[5] R. Szeliski, *Computer Vision: Algorithms and Applications*. Springer, 2010.

[6] X. Xu, X. Xie, and Q. Dai, "Real-time 3D video synthesis from binocular stereo camera," in *3DTV Conference: The True Vision - Capture, Transmission and Display of 3D Video*, May 2008, pp. 133–136.

[7] C. Riechert, F. Zilly, P. Kauff, J. Guther, and R. Schafer, "Fully automatic stereo-to-multiview conversion in autostereoscopic displays," in *The best of IET and IBC*, vol. 4, Sep 2012, pp. 8–14.

[8] F. Seitner, M. Nezveda, M. Gelautz, G. Braun, C. Kapeller, W. Zellinger, and B. Moser, "Trifocal system for high-quality inter-camera mapping and virtual view synthesis," in *2015 International Conference on 3D Imaging (IC3D)*, Dec 2015, pp. 1–8.

[9] M. Dumont, *Real-Time View Interpolation for Eye Gaze Corrected Video Conferencing*. PhD thesis, Hasselt University, Belgium, 2015.

[10] C.-K. Liao, H.-C. Yeh, K. Zhang, V. Geert, T.-S. Chang, and G. Lafruit, "Stereo matching and viewpoint synthesis FPGA implementation," in *3D-TV System with Depth-Image-Based Rendering*, New York, USA: Springer, 2013, pp. 69–106.

[11] A. Akin, R. Capoccia, J. Narinx, J. Masur, A. Schmid, and Y. Leblebici, "Real-time free viewpoint synthesis using three-camera disparity estimation hardware," in *2015 IEEE International Symposium on Circuits and Systems (ISCAS)*, May 2015, pp. 2525–2528.

[12] L. Zhang, K. Zhang, T. S. Chang, G. Lafruit, G. K. Kuzmanov, and D. Verkest, "Real-time high-definition stereo matching on FPGA," in *Proceedings of the 19th ACM/SIGDA International Symposium on Field Programmable Gate Arrays*. ACM, 2011, pp. 55–64.

[13] Y. Shan, Y. Hao, W. Wang, Y. Wang, X. Chen, H. Yang, and W. Luk, "Hardware acceleration for an accurate stereo vision system using mini-census adaptive support region," *ACM Trans. Embed. Comput. Syst.*, vol. 13, no. 4s, pp. 132:1–132:24, Apr. 2014.

[14] M. Jin and T. Maruyama, "Fast and accurate stereo vision system on FPGA," *ACM Trans. Reconfigurable Technol. Syst.*, vol. 7, no. 1, pp. 3:1–3:24, 2014.

[15] W. Wang, J. Yan, N. Xu, Y. Wang, and F. H. Hsu, "Real-time high-quality stereo vision system in FPGA," *IEEE Trans. Cir. and Sys. for Video Technol.*, vol. 25, no. 10, pp. 1696–1708, 2015.

[16] Y. Li, C. Yang, W. Zhong, Z. Li, and S. Chen, "High throughput hardware architecture for accurate semi-global matching," in *22nd Asia and South Pacific Design Automation Conference (ASP-DAC)*, Jan 2017, pp. 641–646.

[17] H. J. Chen, F. H. Lo, F. C. Jan, and S. D. Wu, "Real-time multi-view rendering architecture for autostereoscopic displays," in *Proceedings of 2010 IEEE International Symposium on Circuits and Systems*, May 2010, pp. 1165–1168.

[18] Y. K. Lai, Y. C. Chung, and Y. F. Lai, "Hardware implementation for real-time 3D rendering in 2D-to-3D conversion," in *2013 IEEE International Symposium on Circuits and Systems (ISCAS2013)*, May 2013, pp. 893–896.

[19] P. F. Jin, S. J. Yao, D. X. Li, L. H. Wang, and M. Zhang, "Real-time multi-view rendering based on FPGA," in *2012 International Conference on Systems and Informatics (ICSAI2012)*, May 2012, pp. 1981–1984.

[20] J. Wang and L. A. Rønningen, "Real time believable stereo and virtual view synthesis engine for autostereoscopic display," in *2012 International Conference on 3D Imaging (IC3D)*, Dec 2012, pp. 1–6.

[21] D. Scharstein and R. Szeliski, "A taxonomy and evaluation of dense two-frame stereo correspondence algorithms," *International Journal of Computer Vision*, vol. 47, no. 1, pp. 7–42, 2002.

[22] H. Hirschmüller and D. Scharstein, "Evaluation of stereo matching costs on images with radiometric differences," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 31, no. 9, pp. 1582–1599, Sept 2009.

[23] G. Führ, G. P. Fickel, L. P. Dal'Aqua, C. R. Jung, T. Malzbender, and R. Samadani, "An evaluation of stereo matching methods for view interpolation," in *2013 IEEE International Conference on Image Processing*, Sept 2013, pp. 403–407.

[24] N. Y.-C. Chang, T.-H. Tsai, B.-H. Hsu, Y.-C. Chen, and T.-S. Chang, "Algorithm and architecture of disparity estimation with mini-census adaptive support weight," *IEEE Trans. Cir. and Sys. for Video Technol.*, vol. 20, no. 6, pp. 792–805, Jun. 2010.

[25] W. S. Fife and J. K. Archibald, "Improved census transforms for resource-optimized stereo vision," *IEEE Trans. Cir. and Sys. for Video Technol.*, vol. 23, no. 1, pp. 60–73, Jan 2013.

[26] F. Tombari, S. Mattoccia, and L. Di Stefano, "Segmentation-based adaptive support for accurate stereo correspondence," in *Proceedings of the 2Nd Pacific Rim Conference on Advances in Image and Video Technology*, 2007, pp. 427–438.

[27] S. Rogmans, J. Lu, P. Bekaert, and G. Lafruit, "Real-time stereo-based view synthesis algorithms: A unified framework and evaluation on commodity GPUs," *Signal Processing: Image Communication*, vol. 24, no. 1, pp. 49 – 64, 2009.

[28] C. Vázquez, W. J. Tam, and F. Speranza, "Stereoscopic imaging: filling disoccluded areas in depth image-based rendering," in *Proc. SPIE*, vol. 6392, 2006, p. 63920D.

[29] J. Overes, *Occlusion filling in depth-image-based rendering*. Master thesis, Delft University of Technology, The Netherlands, 2009.

[30] K. Zhang, J. Lu, Q. Yang, G. Lafruit, R. Lauwereins, and L. V. Gool, "Real-time and accurate stereo: A scalable approach with bitwise fast voting on CUDA," *IEEE Trans. Cir. and Sys. for Video Technol.*, vol. 21, no. 7, pp. 867–878, July 2011.

[31] M. A. Vega-rodríguez, J. M. Sánchez-pérez, and J. A. Gómez-pulido, "An FPGA-based implementation for median filter meeting the real-time requirements of automated visual inspection systems," in *Proceedings of the 10th Mediterranean Conference on Control and Automation*, Lisbon, 2002.

[32] Z. Wang, A. C. Bovik, H. R. Sheikh, and E. P. Simoncelli, "Image quality assessment: from error visibility to structural similarity," *IEEE Transactions on Image Processing*, vol. 13, no. 4, pp. 600–612, April 2004.

[33] C. Ttofis, C. Kyrkou, and T. Theocharides, "A low-cost real-time embedded stereo vision system for accurate disparity estimation based on guided image filtering," *IEEE Transactions on Computers*, vol. 65, no. 9, pp. 2678–2693, Sept 2016.

[34] C. Ttofis and T. Theocharides, "Towards accurate hardware stereo correspondence: A real-time FPGA implementation of a segmentation-based adaptive support weight algorithm," in *Proc. Design Autom. Test Eur. Conf. Exhibit. (DATE)*, 2012, pp. 703–708.

[35] S. Jin, J. Cho, X. D. Pham, K. M. Lee, S. K. Park, M. Kim, and J. W. Jeon, "FPGA design and implementation of a real-time stereo vision system," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 20, no. 1, pp. 15–26, Jan 2010.

[36] L. Zhang and W. J. Tam, "Stereoscopic image generation based on depth images for 3D TV," *IEEE Transactions on Broadcasting*, vol. 51, no. 2, pp. 191–199, June 2005.

[37] T. C. Yang, P. C. Kuo, B. D. Liu, and J. F. Yang, "Depth image-based rendering with edge-oriented hole filling for multiview synthesis," in *2013 International Conference on Communications, Circuits and Systems (ICCCAS)*, vol. 1, Nov 2013, pp. 50–53.

[38] Z. Sun and C. Jung, "Real-time depth-image-based rendering on GPU," in *2015 International Conference on Cyber-Enabled Distributed Computing and Knowledge Discovery*, Sept 2015, pp. 324–328.

[39] P. J. Lee and Effendi, "Nongeometric distortion smoothing approach for depth map preprocessing," *IEEE Transactions on Multimedia*, vol. 13, no. 2, pp. 246–254, April 2011.

[40] K.-H. Chen, C.-H. Chen, C.-H. Chang, J.-Y. Liu, and C.-L. Su, "A shape-adaptive low-complexity technique for 3D free-viewpoint visual applications," *Circuits, Systems, and Signal Processing*, vol. 34, no. 2, pp. 579–604, Feb 2015.

[41] P.-C. Kuo, J.-M. Lin, B.-D. Liu, and J.-F. Yang, "High efficiency depth image-based rendering with simplified inpainting-based hole filling," *Multidimensional Systems and Signal Processing*, vol. 27, no. 3, pp. 623–645, Jul 2016.