IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Received 24 May 2022, accepted 29 June 2022, date of publication 7 July 2022, date of current version 15 July 2022.

Digital Object Identifier 10.1109/ACCESS.2022.3189008

RESEARCH ARTICLE

A High-Speed Low-Cost Hardware Implementation for Depth Estimation Using Disparity Fusion Method

YOU-RONG CHEN[1], WEI-TING CHEN[2], SHAO-CHIEH LIAO[1], PEI-YIN CHEN[1], (Senior Member, IEEE), HONG-YU FANG[2], AND TZU-YOU TAI[3]

[1]Digital IC Design Laboratory, Department of Computer Science and Information Engineering, National Cheng Kung University, Tainan 70101, Taiwan
[2]Novatek Microelectronics Corporation, Hsinchu 30076, Taiwan
[3]MediaTek Inc., Hsinchu 30078, Taiwan

Corresponding author: Pei-Yin Chen (pychen@csie.ncku.edu.tw)

This work was supported in part by the Ministry of Science and Technology, Taiwan, under Grant 110-2622-8-006-004-TA.

ABSTRACT Depth estimation using stereo images can be achieved by calculating the disparity values between the left and the right images captured by two parallel cameras. Reconstructing depth information from 2D images is crucial in many applications, such as self-driving vehicles and robot navigation. Furthermore, most of these applications are employed with resource-constrained devices and have real-time requirements. In this paper, a high-speed, low-cost hardware implementation for disparity estimation is proposed. We adopted the novel disparity fusion method in our architecture, which can significantly reduce the number of calculations in the overall process. A refinement method is also designed to reduce the error rate of the resulting depth map and improve the tolerance to light noise. The proposed algorithm was implemented with the Kintex-7 field-programmable gate array. Its performance was tested by using the Middlebury-Version 2 and -Version 3 datasets. The proposed algorithm provides an operating speed of 118 fps with an error rate of only 6.36%. Compared with other state-of-the-art designs used for similar applications, the proposed method can achieve a 34.6% reduction in the error rate while providing the highest speed with competitive hardware cost.

INDEX TERMS Depth estimation, hardware implementation, high-speed, low-cost, stereo matching.

I. INTRODUCTION

Depth information has been widely used in many applications, such as for self-driving vehicles and robot navigation. Disparity-to-depth conversion is the simplest method by which to obtain depth information. Given two rectified images captured by two horizontal parallel cameras, a pixel at coordinate ($x1$, y) in the left image can find a corresponding pixel at the same vertical coordinate y in the right image. Let the corresponding pixel be at coordinate ($x2$, y), the disparity value is the absolute distance between $x1$ and $x2$. An object with larger disparity values implies that it is closer to the camera. As a result, the depth precision is closely related to the accuracy of the estimated disparity values. Besides, the disparity estimation process must be fast enough to meet real-time requirements for the above-mentioned applications. Using multicore CPUs or GPUs may be able to speed up processing, but their high resource costs and complex computing characteristics will also limit their range of application, which makes hardware implementation on field-programmable gate arrays (FPGAs) or application-specific integrated circuits (ASICs) a better solution. To apply the disparity estimation circuit in end-user equipment, its design has to be low complexity and with low power consumption. The disparity estimation algorithm also needs to deal with light noise problem, since the captured images from end-user equipment may vary considerably in brightness.

In this paper, a hardware-based disparity estimation algorithm is proposed. Compared with other state-of-the-art hardware designs, our design has the highest operating speed and lowest bad pixel rate with competitive resource usage. The main contributions are listed as follows.

The associate editor coordinating the review of this manuscript and approving it for publication was Ilaria De Munari.

(1) A novel disparity fusion method is proposed. By adopting the disparity fusion method, the amount of computation of the disparity estimation algorithm can be significantly reduced.

(2) Taking benefits from the disparity fusion method, the resource usage of each part of the hardware implementation for the disparity estimation algorithm is reduced. The operating speed of the hardware is also improved, which can reach real-time requirements.

(3) A novel continuous plane refinement method is proposed to improve the quality of the resulting depth maps and enhance the tolerance of the proposed algorithm to brightness change. The proposed algorithm can still have a good effect on the images with light noise.

The rest of the paper is organized as follows: Section II describes a few well-known disparity estimation algorithms. The main ideas of our proposed method are described in Section III. Section IV introduces the proposed algorithm in detail. The hardware architectures of the proposed algorithm are presented in Section V. Section VI lists the experimental results and compares them with the results of state-of-the-art works. Finally, the conclusion is provided in Section VII.

II. RELATED DISPARITY ESTIMATION ALGORITHMS

The disparity estimation algorithms can be divided into two categories: global matching algorithm and regional matching algorithm. The following discussion provides an analysis of the two different algorithms.

A. GLOBAL MATCHING ALGORITHM

The global matching algorithm takes the entire image when calculating the disparity values and thus achieves better estimation accuracy. The energy function is usually adopted to find the best disparity, which generates the least energy. The energy function consists of two items: The first is the similarity item, where a more similar pixel leads to a smaller energy value. The second is the gradient of smoothing between the central pixel and the surrounding pixels, where a smoother gradient leads to a smaller energy value. There are several common disparity estimation methods that use the concept of global matching algorithm, such as dynamic programming [1], belief propagation [2], graph cuts [3]. However, even though global matching algorithms can lower the error rates, they are usually time-consuming. Although some of them use multicore CPUs or GPUs to speed up processing [4]–[6], their high resource costs and complex computing characteristics make them unsuitable for embedded systems such as digital signal processors (DSPs), mobile and automotive electronics.

B. REGIONAL MATCHING ALGORITHM

To reduce costs and improve operating speed, regional matching algorithms [7]–[15], [17]–[20], which use a fixed-sized sliding window to replace the inspection of the whole image during operation, have been proposed. Because they have low-complexity computing features, regional matching algorithms are especially suitable for implementation in hardware such as DSPs, FPGAs, and ASICs [7], [8]. However, the error rates are increased in spite of the fact that their operating speed of hardware implementation can meet real-time requirements. Therefore, several works have focused on achieving higher frames per second (fps) with acceptable precision.

Adaptive support weight (ADSW) methods [9]–[15], which can effectively improve precision, have become the mainstream methods for hardware implementations. In ADSW-based methods, the initial energy and the corresponding weights are calculated first. The previous energy values and weights are used to calculate the new energy value, and then the disparity value is determined based on the new energy value. The difference in color and distance determines the weight calculation, where a larger difference indicates a smaller corresponding weight, and vice versa. Although the operation of ADSW is simple, it is necessary to recalculate the weight and the sum of multiplications whenever a new energy value is calculated. Thus, the ADSW methods have relatively high memory usage requirements and significant resource consumption, which may cause a bottleneck in hardware implementation.

Some studies have recently adopted the guided image filter (GIF) method [16] in their architectures to reduce area costs and improve operating speeds. For example, Hosni *et al.* [17] proposed a real-time method that obtained improved performance by replacing the bilateral filters used in the ADSW method with GIFs. Ttofis *et al.* [18], [19] introduced the parallel stereo matching FPGA architecture based on GIFs. Vala *et al.* [20] optimized the architecture by exploiting discrete wavelet transform (DWT) technology in order to significantly reduce the required hardware resources with only a small sacrifice in the error rate. Although GIF-based algorithms appear to be a suitable solution for the disparity estimation issue, their performance worsens when the application scenarios become more complex, or the scenarios are impacted by light noise.

To improve the accuracy of estimation in complex and light-noise environments, some algorithms [23] involved the use of encoding methods to calculate the energy or the cost and then decided the disparity results. However, to the best of our knowledge, there are still many problems that need to be overcome when processing these environments. Thus, designing an algorithm with light noise resistance under low-cost and high-speed conditions is an important and urgent issue.

We proposed a GIF-based design which adopts the novel disparity fusion method. The computation complexity can be significantly reduced and the operating speed of our design can reach real-time requirements. A refinement method is also proposed, with which our algorithm can enhance its output quality and tolerance to light noise.

III. MAIN IDEAS

A. THE LOCAL MINIMUM

To generate the disparity value for each point in the disparity map, the pixel in the left image has to locate its most similar pixel in the right image. Since the lenses are placed in parallel, the pair of corresponding points in the left and the right images will be on the same horizontal line. The disparity estimation algorithm will calculate the difference between the left and the right images with each disparity value within the disparity range and obtain the cost value of them. The cost value represents the similarity between the left and the right images, where a lower difference corresponds to a lower cost.

It is assumed that if the lowest cost value of the pixel at coordinate (i, j) occurs when the disparity value is k, then its cost value will increase as the absolute value between the given disparity value and k increases. For example, when the point $P_r(i-k, j)$ in the right image is the most similar point to $P_l(i, j)$ in the left image. This thus leads to the lowest calculated cost. In addition, one of the points $P_r(i-(k-1), j)$ and $P_r(i-(k+1), j)$ should get the second-lowest-cost value. In the smooth region of image, the pixel values tend to change progressively. As a result, the hypothesis holds when the given disparity value is close to the disparity value with the lowest cost and no edge occurs around the processing pixel.

Based on the above assumption, the number of cost maps can be reduced. Even if the disparity value that leads to the lowest cost is skipped, the found disparity value will still be close to it. With the cost maps generating from the partial disparity range, the proposed disparity fusion method can generate disparity map that covers the entire disparity range. For example, we can calculate the left cost maps with only even-order disparity values, and calculate the right cost maps with only odd-order disparity values. The left cost maps and the right cost maps each cover half of the disparity range, and the calculating complexity is halved. And then we can merge the left cost maps and right cost maps to get consistent disparity map with the proposed disparity fusion method. The disparity values in the resulting consistent disparity map are thus able to cover the entire disparity range.

B. THE CONTINUOUS PLANE

Each contiguous region bounded by enclosing edges is assumed to have the same depth value in the depth map. Then, the disparity values of these points should also be the same. On the other hand, when the points of a contiguous region in 3D space are mapped to the 2D plane, they will be on a continuous plane. Based on the above analysis, the disparity values should be the same in the continuous plane.

IV. PROPOSED DISPARITY ESTIMATION ALGORITHM

To be suitable for hardware implementation, the proposed algorithm is region based. Fig. 1 shows the overview of the proposed disparity estimation algorithm. It is depicted on a hardware basis to describe the parallelism of the circuit processing flow. As shown in Fig. 1, the input of the proposed

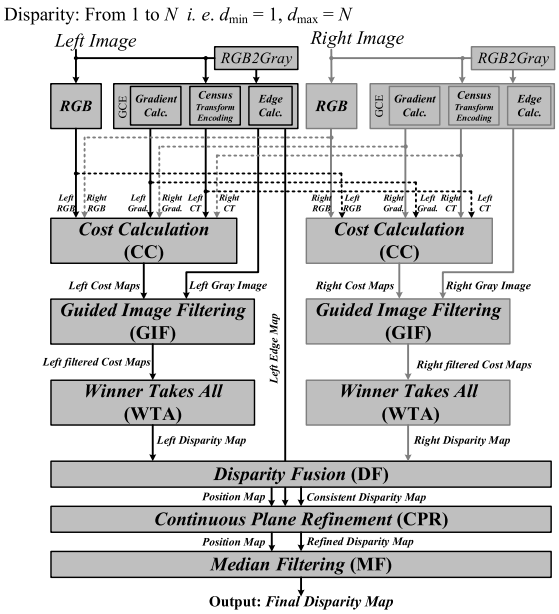

FIGURE 1. Hardware-based overview of the proposed disparity estimation algorithm.

algorithm is one pair of RGB images from two horizontal parallel lenses. Hence, one point in the left image should be on the same horizontal line of the right image. The only difference is that there is one disparity value (d) between them. The disparity range (N) is set as 64, and the disparity values will be between 1 to 64.

The proposed algorithm consists of six main steps: The first step is essential information generation and cost calculation (CC). One disparity value generates one corresponding cost map. Then, all of the cost maps are filtered by the guided image filter (GIF) in the second step. In the third step, the disparity value with the corresponding lowest cost in each point is set by using the winner takes all (WTA) operation, and the initial disparity map is constructed. After finishing the first three steps, the left and the right disparity maps are generated, respectively. The fourth step is to merge the left and the right disparity maps into the consistent disparity map with the proposed disparity fusion (DF) method. The fifth step is responsible for refining the consistent disparity map. The continuous plane refinement (CPR) process is designed to refine the consistent disparity map and improve its tolerance to light noise. Finally, the refined disparity map is filtered by the median filter (MF) and output as the final result. It is worth noting that the calculation complexity and the hardware resource usage of each disparity map generation step, which are CC, GIF, and WTA, are all significantly reduced by adopting the DF method, since it can reduce the number of requiring cost maps for generating disparity maps. The following subsections give a detailed introduction to each step.

A. ESSENTIAL INFORMATION GENERATION AND COST CALCULATION

There are four kinds of essential information used for the cost map generation. Two of them must convert the input

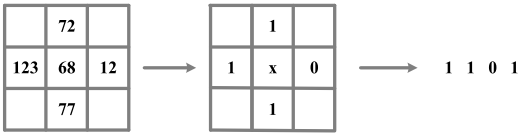

FIGURE 2. The processing of census transform encoding.

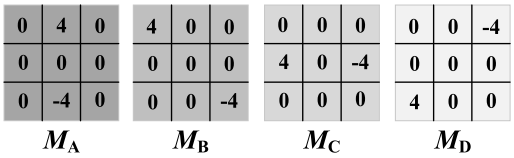

FIGURE 3. Four different angular kernels, named M_A, M_B, M_C, and M_D.

RGB image to grayscale before calculation. The following discussion introduces how to generate the information.

The first kind of information is RGB data, which can be obtained by directly disassembling the input image into red, green, and blue channels. The second kind of information, gradient data, can be generated by doing the convolution operation with the grayscale image and the Sobel operators. The third kind of information is the census transform encoding, which is a special coding method. It first compares the pixel values of the surrounding points with the measured point. If the pixel value of the surrounding point is greater than the pixel value of the measured point, the comparison result is set as 1. Otherwise, it is set as 0. After all the comparison results are generated, they are encoded into one string of characters. Take Fig. 2 as an example. The value of the measured pixel is 68, and the values of the surrounding points are 72, 123, 12, and 77. After the census transform encoding operation, the encoded string of characters is 1101.

The last kind of information is edge information. Four 3×3 kernels with different angles are taken to find the edge information. As shown in Fig. 3, the angle of each kernel is 0° (M_A), 45° (M_B), 90° (M_C), and 135° (M_D), respectively. The proposed algorithm selects the maximum absolute value among four convolution results for the edge judgment. As shown in (1), assume that the $I(i,j)$ is the processed point in the RGB image. Then $|M_A * I(i,j)|$, $|M_B * I(i,j)|$, $|M_C * I(i,j)|$, and $|M_D * I(i,j)|$ are the absolute values of the convolution results based on the angular kernels. And T_{max} is the maximum value among them.

$$T_{max} = \max\left(|M_A * I(i,j)|, |M_B * I(i,j)|, |M_C * I(i,j)|, |M_D * I(i,j)|\right). \quad (1)$$

Then, the edge is determined according to the conditions in (2). The first condition is that when T_{max} is greater than or equal to the threshold T_{edge}, then $I(i,j)$ is regarded as located at the edge. The second condition is that $I(i,j)$ is at the border of the RGB image, then it is regarded as located directly at the edge. After all of the points are checked, the edge map (*EdgeMap*) is produced.

$$EdgeMap(i,j) = \begin{cases} 1, & if\ T_{max} \geq T_{edge} \\ 1, & if\ i = 0\ or\ width - 1 \\ 1, & if\ j = 0\ or\ height - 1 \\ 0, & otherwise. \end{cases} \quad (2)$$

After all four kinds of essential information have been generated, the cost map can be generated with cost calculation (CC) process, where one disparity value corresponds to one cost map. Hence, the total number of cost maps depends on how many disparity values the left and the right images check. The operation of CC consists of three parts: color cost (COLC), gradient cost (GRDC), and Hamming distance cost (HDC). These three costs are calculated by using six items of information: the RGB data of the left and the right images, the gradient data of the left and the right images, and the census transform encoding of the left and the right images. The following discussion describes a detailed analysis of how these three cost parts are calculated.

The first part of the cost is the color cost (COLC), which represents the truncated absolute difference between the RGB data in the left image and the right image. Assume that I_l and I_r represent the left image and the right image. I_l^R, I_l^G, and I_l^B are the pixel values of the red, green, and blue channels in the left image, and I_r^R, I_r^G, and I_r^B are the same values in the right image. The point p is the pixel being processed; d_l and d_r are the disparity values, and M_l and M_r are, respectively, the COLCs of the left image and the right image.

$$M_l(p, d_l) = \frac{1}{3} \times \sum\nolimits_{i=R,G,B} |I_l^i(p) - I_r^i(p - d_l)|, \quad (3.1)$$

$$M_r(p, d_r) = \frac{1}{3} \times \sum\nolimits_{i=R,G,B} |I_l^i(p + d_r) - I_r^i(p)|. \quad (3.2)$$

Formulas (3.1) and (3.2) are used to calculate the COLCs of the left and the right images. It can be seen that the COLC can be obtained after the results of the three channels are added and averaged.

The second part of the cost is the gradient cost (GRDC), which represents the truncated absolute difference in the gradient. Assume that Y_l and Y_r are the grey images of I_l and I_r. ∇x and ∇y are, respectively, the Sobel kernels of the x and y axes, and G_l and G_r are, respectively, the GRDCs of the left and the right images.

$$G_l(p, d_l) = |\nabla_x(Y_l(p)) - \nabla_x(Y_r(p - d_l))| + |\nabla_y(Y_l(p)) - \nabla_y(Y_r(p - d_l))|, \quad (4.1)$$

$$G_r(p, d_r) = |\nabla_x(Y_l(p + d_r)) - \nabla_x(Y_r(p))| + |\nabla_y(Y_l(p + d_r)) - \nabla_y(Y_r(p))|. \quad (4.2)$$

Formulas (4.1) and (4.2) are used to calculate the GRDCs for the left and the right images. It can be seen that the GRDC can be obtained after the absolute difference in the gradient for the x and y axes are added together.

The last part of the cost is the *Hamming distance cost* (HDC), which represents the difference in the census transform encoding. To calculate the HDC, the census transform

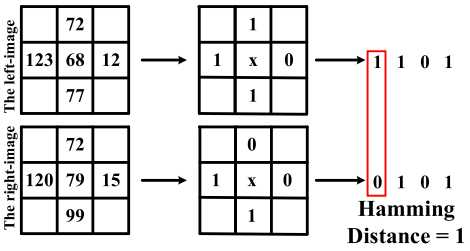

FIGURE 4. Operation of HDC.

encoding string generated from the left and the right images are checked, where the HDC is the number of differences in the character. Assume that H_l and H_r are, respectively, the HDCs of the left and right images, where the *HammingDistance* (*HD*) represents the process that calculates the distance between two strings of census transform encoding. The HDC formula can be represented as:

$$H_l\left(p, d_l\right) = HD\left(\text{Census}\left(Y_l, p\right), \text{Census}\left(Y_r, p - d_l\right)\right), \tag{5.1}$$

$$H_r\left(p, d_r\right) = HD\left(\text{Census}\left(Y_l, p + d_r\right), \text{Census}\left(Y_r, p\right)\right). \tag{5.2}$$

The *HD* processing is shown in Fig. 4. Take the process on the left image as an example. Assume the census transform encoding result for point p in the left image, (Census(Y_l, p)), is 1101, and the census transform encoding result for point p-d_l in the right image (Census($Y_r, p-d_l$)) is 0101. These two strings are compared. If the comparison result in any position is different, the value of that position is set as 1. Otherwise, the value of that position is set as 0. After the comparison, the value of each position is set as 1, 0, 0, and 0. Then, the result for the *HD* of point p in the left image is the sum of these values, which is equal to 1.

After the COLCs, GRDCs, and HDCs are all generated, each of them is first compared with the threshold values T_c, T_g, and T_h. When the generated value is greater than the threshold value, it is replaced with the threshold value. This step is to avoid the overflow situation in hardware implementation with limited storage space when the calculated value is too large. Assume that α, β, and γ are the ratio of the COLC, GRDC, and HDC to the total cost. On the basis of experimental tests, $\{\alpha, \beta, \gamma, T_c, T_g, T_h\}$ are set as $\{0.1, 0.7, 0.2, 11, 2, 4\}$. C_l and C_r are, respectively, the produced cost of the left and the right images. The formula of the total cost can be represented as:

$$\begin{aligned} C_l\left(p, d_l\right) = {} & \alpha \cdot \min\left(T_c, M_l\left(p, d_l\right)\right) \\ & + \beta \cdot \min\left(T_g, G_l\left(p, d_l\right)\right) \\ & + \gamma \cdot \min\left(T_h, H_l\left(p, d_l\right)\right), \end{aligned} \tag{6.1}$$

$$\begin{aligned} C_r\left(p, d_r\right) = {} & \alpha \cdot \min\left(T_c, M_r\left(p, d_r\right)\right) \\ & + \beta \cdot \min\left(T_g, G_r\left(p, d_r\right)\right) \\ & + \gamma \cdot \min\left(T_h, H_r\left(p, d_r\right)\right). \end{aligned} \tag{6.2}$$

After cost calculation process, the cost maps whose number is equal to the number of given disparity values to the left and the right images can be successfully generated. It is worth noting that the number of given disparity values to the left and the right images should be equal to the disparity range in general. When the disparity range is set as 64, 128 cost maps will be generated. However, the number of given disparity values can be reduced with the proposed disparity fusion method, which is able to utilize information from the left cost maps and the right cost maps to cover the entire disparity range.

B. GUIDED IMAGE FILTER

Since the cost values are calculated separately for each point during the cost map generation process, each value in the cost map is independent. Without referring to the characteristics of the surrounding points, the generated value is prone to noise, especially when the calculation blocks are at the edges of objects. The resulting noise may raise the error rate of the estimation results. Thus, the guided image filter (GIF) is responsible for filtering the noise in the cost map. The GIF, which can smooth the image while keeping its edge clear, was first proposed in [16]. It uses input image I as the guidance image to filter the cost map c. Assume that the ω_k is a square processing window centered at pixel k. Its radius is r, and its side length is $2r + 1$. The relationship among each point in the filtered cost map q_i, an initial cost map c_i, and noise n_i is represented as:

$$q_i = c_i - n_i \quad \forall i \in \omega_k. \tag{7}$$

On the other hand, in the GIF method proposed in [16], it is assumed that the point in the filtered cost map (q_i) is generated by a linear transform model of the point in the guidance image (I_i). Assume that a_k and b_k are both the linear coefficients. The relationship among q_i and I_i is represented as:

$$q_i = a_k * I_i + b_k \quad \forall i \in \omega_k. \tag{8}$$

Since the desired number (q_i) in (7) and (8) are equal, the noise can be minimized by using these two formulas. Assuming that ε is the legalization parameter to prevent a_k from becoming too large, the noise can be minimized as:

$$argmin(\Sigma_{i\in\omega_k}\left((a_k * I_i + b_k - c_i)^2 + \varepsilon * a_k^2\right)). \tag{9}$$

This function can be solved by using the linear ridge regression model proposed in [16]. Here, we directly quote the result derived in [16]. The solution for a_k and b_k are given in (10) and (11), respectively.

$$a_k = \frac{1}{\sigma_k^2 + \varepsilon} \times \frac{\Sigma_{i\in\omega_k} I_i * c_i - \mu_k * \overline{c_k}}{|\omega_k|} = \frac{cov(I, c)}{var\left(I\right) + \varepsilon}, \tag{10}$$

$$b_k = \overline{c_k} - a_k * \mu_k = mean\left(c\right) - a_k * mean(I). \tag{11}$$

μ_k and σ_k^2 respectively represent the mean and variance of the guidance image I in ω_k. $|\omega_k|$ represents the total number of pixels in ω_k. $\overline{c_k}$ is the mean of the cost map c in ω_k.

We can substitute these two linear coefficients a_k and b_k into (8) to obtain the filtered cost value. However, since one pixel may be covered by several overlapping windows, this causes it to lead to different q_i values. This problem can be solved through averaging the overlapping windows. Therefore, Formula (8) can be rewritten as:

$$q_i = \frac{1}{\omega} \sum_{k, i \in \omega_k} (a_k * I_i + b_k) = mean\,(a_k) * I_i + mean\,(b_k)\,, \tag{12}$$

where ω represents the number of windows that contain point i.

Algorithm 1 shows the example of the GIF operation based on (7)-(12). The input consists of one guidance image (I), and one guided cost map (c). The output is the filtered cost map (q). f_{mean} is the mean filter with a window of radius r. *corr*, *var*, and *cov* represent the correlation, variance, and covariance operations, respectively. From (10), we know that to get the value of a_k, it is necessary to calculate both the covariance and variance, which can be generated from the correlation. Likewise, from (11), we know that to get the value of b_k, it is necessary to further calculate *mean* (I) and *mean*(c), representing the mean filtering to the guidance image and the guided cost map. After the GIF operation, filtered cost maps for the left and the right images can be obtained.

Algorithm 1 Algorithm of GIF

01: **INPUT**: guidance image I, guided cost map c
02: **OUTPUT**: filtered cost map q
03: Parameters: $\boldsymbol{\varepsilon}$
04: $mean\,(I) \leftarrow f_{mean}\,(I)$, $mean\,(c) \leftarrow f_{mean}\,(c)$;
05: $corr\,(I) \leftarrow f_{mean}\,(I.*I)$, $corr\,(I,c) \leftarrow f_{mean}\,(I.*c)$;
06: $var\,(I) \leftarrow corr\,(I) - mean\,(I)\,.*mean\,(I)$; // variance of I
07: $cov\,(I,c) \leftarrow corr\,(I,c) - mean\,(I)\,.*mean\,(c)$; // *covariance* (I,c)
08: $a_k \leftarrow cov\,(I,c)\,/(var\,(I) + \boldsymbol{\varepsilon})$;// computing a_k
09: $b_k \leftarrow mean\,(c) - a_k\,.*mean\,(I)$; // computing b_k
10: $mean\,(a_k) \leftarrow f_{mean}\,(a_k)$, $mean\,(b_k) \leftarrow f_{mean}\,(b_k)$;
11: $q \leftarrow mean\,(a_k) * I + mean\,(b_k)$;

C. WINNER TAKES ALL

After the filtered cost maps are generated, the operation must select the suitable disparity values by referencing these cost maps. The *winner takes all* (WTA) processing method is adopted to find the suitable disparity value for each point in order to construct the left and the right disparity maps.

The pseudo code of the WTA is given in Algorithm 2. Considering the hardware cost and the accuracy of resulting depth map, we only use even disparity values to calculate the left cost maps, and only odd disparity values are used to calculate the right cost maps. As a result, there are 32 cost maps generated from the left image and 32 cost maps generated from the right image. The WTA first compares all of the cost maps and then selects one disparity value corresponding to the lowest cost for each point. It is worth noting that the comparison method in the WTA uses the serial comparison method; that is, the preliminary results from the pairwise comparisons are compared with each other again until the final result is obtained. Finally, the disparity values of each point are determined. After the WTA has been applied, the left disparity map (D_l) and the right disparity map (D_r) are produced.

Algorithm 2 Algorithm of Winner Takes All

01: **INPUT**: filtered cost maps $q_{l,d=2,4,6...64}$ $q_{r,d=1,3,5...63}$
02: **OUTPUT**: disparity maps D_l, D_r
03: Parameters: $width$, $height$, tmp_l, tmp_r
04: **for** $j = 0$ to $height - 1$ **do**
05: **for** $i = 0$ to $width - 1$ **do**
06: $tmp_l \leftarrow \infty$, $tmp_r \leftarrow \infty$;
07: **for** $k = 1$ to 32 **do**
08: $d_l \leftarrow 2{\times}k$, $d_r \leftarrow 2{\times}k - 1$;
09: **if** $tmp_l > q_l(i,j,d_l)$ **then**
10: $D_l(i,j) \leftarrow d_l$, $tmp_l \leftarrow q_l(i,j,d_l)$; //WTA operation
11: **if** $tmp_r > q_r(i,j,d_r)$ **then**
12: $D_r(i,j) \leftarrow d_r$, $tmp_r \leftarrow q_r(i,j,d_r)$; // WTA operation
13: **end do**
14: **end do**
15: **end do**

D. DISPARITY FUSION

To reduce the calculation complexity, the number of disparity values used in CC operation is reduced. The values in D_l are all even and those in D_r are all odd. Then the novel disparity fusion method has to be adopted to recover the required disparity information.

According to the hypothesis depicted in Section III-A, we know that although we may not obtain the lowest cost with the selected even/odd disparity values, the second lowest cost can still be obtained. Thus, if one disparity value of $D_l(i,j)$ is k, an even value, then the possible disparity values that are consistent with the corresponding point in the right disparity map are $k-1$, k, and $k+1$. By mutual confirming these three possible disparity values between the left and the right disparity maps, the consistent disparity value can be confirmed.

The operation of the proposed disparity fusion method is shown in Algorithm 3. Under the condition that the interval between given disparity values is one, there are three judgment conditions. All judgments are based on the left image, and consistency is confirmed with the right image. Assume that k is the disparity value of $D_l(i,j)$. The first condition confirms that if the point (i,j) has the disparity value $k-1$, where the criterion is that the disparity value of $D_r(i-(k-1),j)$ is equal to $k-1$. The second condition checks the consistency with the disparity value k. Because k is an even value, if k is the disparity value with the lowest cost, then the disparity value of the corresponding point $D_r(i-k,j)$ must be $k-1$ or $k+1$. Thus, if the absolute difference in the disparity value between $D_l(i,j)$ and $D_r(i-k,j)$ is 1, the consistent disparity value of point (i,j) is k. The last condition confirms that if the point (i,j) has the disparity value $k+1$. The criterion is that the disparity value of $D_r(i-(k+1),j)$

is equal to $k+1$. If the point (i,j) does not satisfy any of the above-mentioned judgment conditions, it is considered an inconsistent point. The disparity values of inconsistent points will be refined in the subsequent steps. After the operation of the disparity fusion, the confirmed disparity map (D_{conf}) can be obtained. The position map (S), which records whether the points in the confirmed disparity map are consistent, is also generated.

Algorithm 3 Algorithm of the Disparity Fusion Method

```
INPUT: left and right disparity maps D_l, D_r
OUTPUT: confirmed disparity map D_conf, recorded position map S
Parameters: width, height
for j = 0 to height − 1 do
  for i = 0 to width − 1 do
    k ← D_l(i, j);
    if k − 1 = D_r(i − (k − 1), j) then //fit first condition
      D_conf(i, j) ← k − 1, S(i, j) ← 1;
    else if abs(k- D_r(i − k, j)) ≤ 1 then //fit second condition
      D_conf(i, j) ← k, S(i, j) ← 1;
    else if k + 1 = D_r(i − (k + 1), j) then //fit third condition
      D_conf(i, j) ← k+1, S(i, j) ← 1;
    else then
      D_conf(i, j) ← 0, S(i, j) ← 0; //inconsistence
  end do
end do
```

E. CONTINUOUS PLANE REFINEMENT

Continuous plane refinement (CPR) is responsible for refining D_{conf}. It is based on the idea mentioned in Section III-B, which indicates that the disparity values should be the same in the continuous plane. The CPR operation can be divided into *Statistics*, *Refinement*, and *Filling*.

Statistics: Since the continuous plane can be regarded as the region between two edges, the *Statistics* operation first counts the total number of consistent points (S_p) between two edges and then finds the largest proportion disparity value (d_{ref}). The number of points with disparity value d_{ref} is also recorded as S_{ref}. The edge information is from the left edge map, which has been generated in the essential information generation step. d_{ref} is qualified according to (13). If the value of S_p is greater than the threshold τ_v and the value of (S_{ref}/S_p) is greater than the threshold τ_h, d_{ref} is regarded as the disparity value for refinement in the current plane.

$$\begin{cases} condition1: & S_p > \tau_v \\ condition2: & \dfrac{S_{ref}}{S_p} > \tau_h \end{cases} \tag{13}$$

Refinement: After d_{ref} is found, the disparity values of the inconsistent points in the current plane are replaced with d_{ref}.

Filling: After the entire disparity map has been processed with the *Statistics* and *Refinement* steps, several points may be still inconsistent. The *Filling* operation will assign the nearest consistent disparity value to these points.

The accuracy of the estimation result can be effectively enhanced by adopting continuous plane refinement. Using edge information in the refinement process can also make up for the deficiencies of the hypothesis in Section III-A. After the CPR operation, the final disparity map (D_{final}) can be obtained. Algorithm 4 shows each step in the CPR operation.

Algorithm 4 Algorithm of the Continuous Plane Refinement

```
INPUT: confirmed disparity map D_conf, recorded position map S, edge information
OUTPUT: final disparity map D_final
Parameters: width, height
foreach row in the image do
  foreach pair of edges (Edge_start, Edge_end) do /* Statistics */
    S_p ← total consistent points between the edges
    d_ref ← the largest proportion disparity value
    S_ref ← total points with disparity value d_ref
    If S_p > τ_v and S_ref/S_p > τ_h then /* Refinement */
      for i ∈ Edge_start to Edge_end do
        If S(i, row)== 0 then
          D_final(i, row) ← d_ref, S(i, row) ← 1
        else then
          D_final(i, row) ← D_conf(i, row)
  end do
end do
for j = 0 to height − 1 do /* Filling */
  for i = 0 to width − 1 do
    If S(i, j) == 0 then
      D_final(i, j) ← nearest consistent disparity value
```

F. MEDIAN FILTER

Before the D_{final} is output, the proposed algorithm will let the D_{final} perform vertical median filtering again. Here, we use the serial comparison method, which was mentioned in [21].

V. HARDWARE ARCHITECTURE

This section describes the hardware implementation of the proposed algorithm. Fig. 5 shows an overview of the proposed architecture. The architecture can be divided into seven units. The *gradient, census transform encoding, and edge information calculation unit* (GCEU) is used to produce the essential information. The cost map generation is implemented with the *cost calculation unit* (CCU). The guided image filter in the second step of the proposed algorithm is achieved with the *guided image filter unit* (GIFU). The *winner takes all unit* (WTAU), and *disparity fusion unit* (DFU) are respectively used to fulfill the WTA and DF operations. Finally, the *continuous plane refinement unit* (CPRU) and *median filter unit* (MFU) are used to refine the disparity map. The digital data format of the proposed hardware architecture is fixed-point binary number. The arithmetic units used in the proposed architecture include adders, subtractors, shifters, and multipliers. Notably, most of the multiplication and division operations are implemented with shifters and adders to save hardware resource usage. The fractional part will be discarded directly, and the experimental results show that the resulted error is acceptable. The multipliers are only used in the GIFU for square calculation. With the proposed disparity fusion method, the number of cost maps can be significantly reduced, which can further reduce the use of multipliers in the

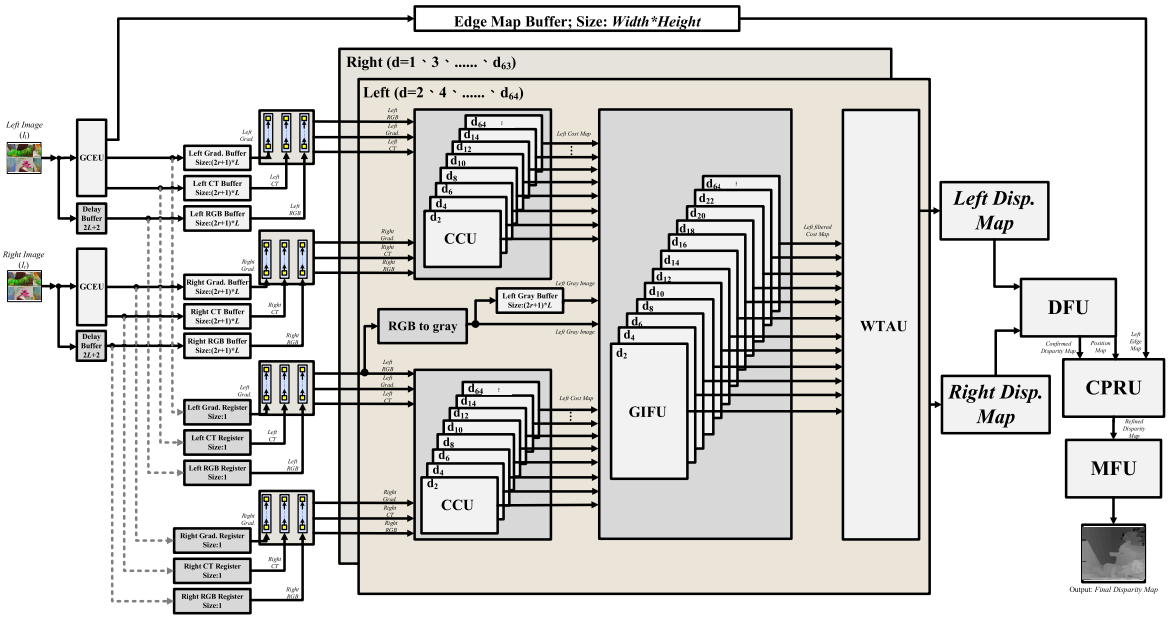

FIGURE 5. The hardware architecture of the proposed algorithm.

GIFU. The following subsections introduce the seven main units in detail.

A. THE GRADIENT, CENSUS TRANSFORM ENCODING, AND EDGE INFORMATION CALCULATION UNIT

As mentioned in Section IV-A, there are four types of essential information for the proposed algorithm. The first kind of information, the RGB data, is concatenated into 24 bits, that is, {R(8 bits), G(8 bits), B(8 bits)}. The RGB data is input into the system and saved in delay buffer. Each piece of image data is input from left to right and from top to bottom. Then, the remaining three types of information are generated.

Fig. 6 shows the details of the GCEU. The gradient data and edge information are calculated with a 3×3 kernel size. The RGB data are first transformed into grey-level data (8 bits) for generating the gradient data and census transform encoding string. The 3×3 kernel takes three rows, each with three pixels, to process the calculation. The hardware implementation takes two lines of block RAM (BRAM) for storing the image data and nine processing registers for performing the convolution. Among the nine processing registers, the first two rows are given by the BRAM, and the last row are directly obtained from the input image or transformed grey-level data. Then, the processing registers will perform convolution with the Sobel kernels or the angular kernels, and the x- and y-axis gradient data and edge information are produced. The census transform encoding result is generated with the transformed grey-level data. The gradient data and the census transform encoding result will then also be saved in delay buffer. The data in delay buffer is required for the operation of the mean filters in the GIFU. With regard to the size of delay buffer, it is determined according to the window size of the mean filter

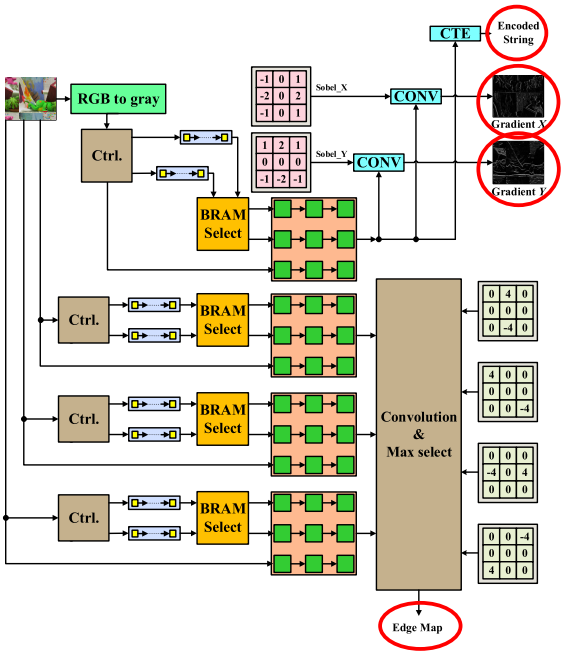

FIGURE 6. The detailed architecture of gradient, census transform encoding, and edge information calculation unit (GCEU).

(r) and the width of the image (L), which will be detailed in Section V-C.

B. THE COST CALCULATION UNIT

The CCU calculates the cost of every point in each cycle and outputs the entire cost map at the same time. The behavior of the CCU is similar for the left and the right images, with the only difference being the selected points. Therefore,

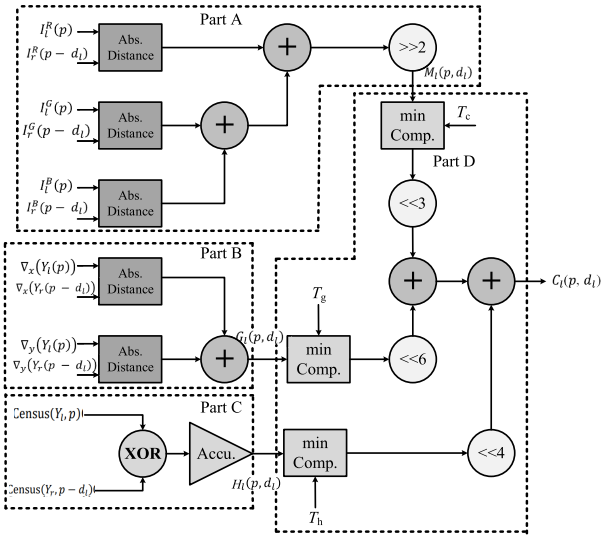

FIGURE 7. The detailed architecture of the CCU (left image calculation is used as an example).

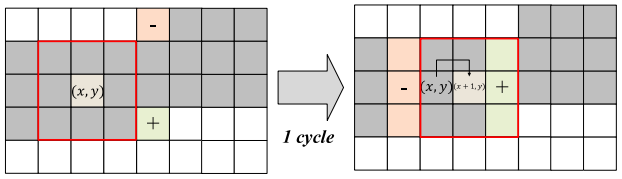

FIGURE 8. The sliding operation of the mean filter.

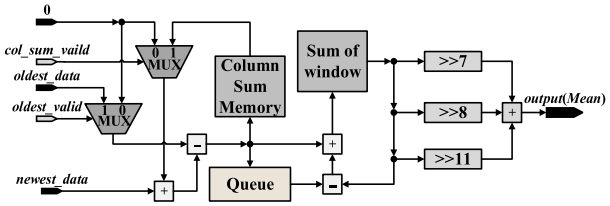

FIGURE 9. The architecture of mean filter.

we only took the operation for the left image as an example. Fig. 7 shows the detailed architecture of the CCU. The cost was calculated by using the disparity value d_l. Part A realizes the COLC operation, that is, (3.1) and (3.2). The difference in the RGB data between the left image and the right image are calculated individually and added together. Notably, the initial formula then divides the result of this addition by "3". However, to reduce hardware costs, we use the shift right 2 bits operation, in which the total is divided by "4", to obtain the approximate result. The difference can be overlooked since the COLC only accounts for a small part of the overall cost. The shift right operation before comparison can also reduce the bit width of the comparator. Since the shift operation can be easily implemented by wiring, the hardware resource usage is further reduced.

Part B realizes the GRDC operation, that is, (4.1) and (4.2). The gradient cost is the sum of the x- and y-axis gradient costs. Finally, Part C is responsible for calculating the HDC according to (5.1) and (5.2). Since the Hamming distance cost records the difference between two strings generated by the census transform encoding, the XOR gate is used to detect the difference and an accumulator is adopted to total the results.

After all partial cost items are generated, part D calculates the total cost, that is, (6.1) and (6.2). The color cost (COLC) is compared with the threshold T_c, and the smaller value of the two is selected. Similarly, the gradient cost (GRDC) and Hamming distance cost (HDC) are compared with T_g and T_h, respectively, and the smaller value of the two is selected. The values of T_c, T_g, and T_h are set as 11, 2, and 4, respectively. These three results are added in a certain proportion to produce the total cost. The COLC ratio is 0.1; the GRDC ratio, 0.7, and the HDC ratio, 0.2. For the hardware implementation, we let the COLC value shift left by 3; the GRDC value, by 6, and the HDC value, by 4. It implies that the COLC, GRDC, and HDC are multiplied by 8, 64, and 16, respectively. The ratios of COLC, GRDC, and HDC are roughly 0.09, 0.73, and 0.18, respectively, which are approximately the same as the parameters used in (6.1) and (6.2).

C. THE GUIDED IMAGE FILTER UNIT

After the CCU generates the cost maps, the GIFU is used to perform the cost map filtering. The input of each GIFU consists of one guidance image I, a guided cost map c, where the output of the GIFU is one filtered cost map q. Here, we only take one GIFU from the operation on the left image as an example to show the design process. Before introducing the GIFU architecture, we first introduce the architecture of the mean filter since the GIFU is constructed of several mean filters.

To perform the mean filtering with a sliding window in GIFU, the newest data must be input for the purpose of renewal, and the oldest data must be discarded. As shown in Fig. 8, the newest data are included in the window, and the oldest data are discarded in each cycle. However, to simplify the architecture, the operation can be replaced by dropping out the most senior column sum and including the newest column sum. The operation is shown in the right part of Fig. 8 with a window of radius of 1. Fig. 9 shows the architecture of the mean filter. The radius of the sliding window adopted in our design was 4, that is, $r = 4$. The *column sum memory* (CSM) subunit is used to save the sum of the data in a single column; its size is equal to the width of the image (L). *col_sum_valid* is not enabled until the first L data are input, and *oldest_valid* is not enabled until $(2r + 1) \times L$ data are input. The first L data are saved into CSM directly, and the $L + 1$ to $(2r + 1) \times L$ data are added together with the initial result saved in the CSM, and then saved back. After *oldest_valid* is enabled, the new data are totalled with the result stored in CSM; the oldest data are removed, and the result is saved back. *Queue* is a bidirectional BRAM that is used to save the latest result from CSM and remove the

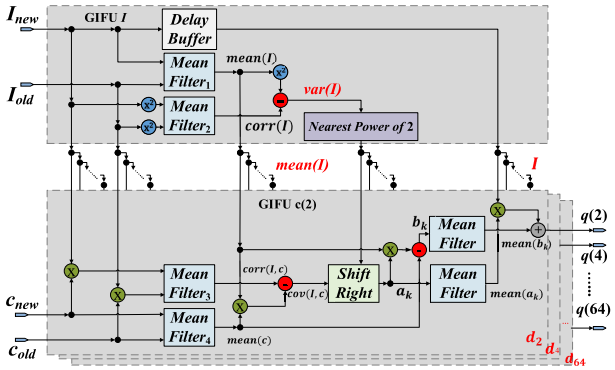

FIGURE 10. The hardware architecture of one GIFU.

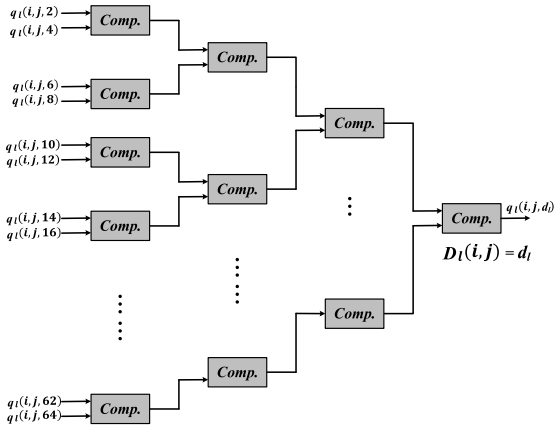

FIGURE 11. The hardware architecture of WTAU.

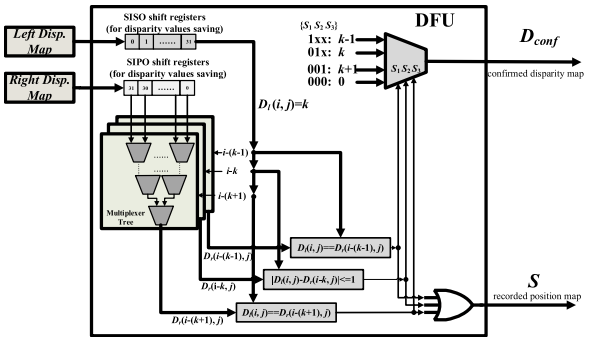

FIGURE 12. The hardware architecture of DFU.

oldest result. The size of *Queue* is $2r + 1$. The *sum of window* (SoW) is a register used to save the mean filter's sum of data. The last step is to take the average of the value saved in SoW. The hardware implementation involves letting the value saved in SoW shift right by 7, 8, and 11 bits, respectively. These three results are totalled to approximate the result of the value in SoW divided by 81. Then the final mean value is obtained.

The GIFU input consists of one guidance image I and several guided cost maps c. Since we only use even order disparity values for the generation of the left cost maps, there are 32 cost maps to be filtered by the GIFU for the left image. I is the original grey image, and the information from the guidance image I is shared by every guided cost map $c(n)$, where n can be 2, 4, 6, ..., 64.

As shown in Fig. 10, the top unit shows the architecture of GIFU I. The input of GIFU I is the newest and oldest grey data, i.e., the guidance image. The delay buffer is used to save the guidance image, which will be processed later. The *mean filter*$_1$ obtains *mean*(I). The square of the guidance image is then filtered by using the *mean filter*$_2$ to obtain the correlation of the guidance image, i.e., *corr*(I). Subsequently, the square of *mean*(I) is subtracted from *corr*(I) to obtain the variance of the guidance image, i.e., *var*(I). Then, guidance image I and the computation results *mean*(I) and *var*(I) are shared with every GIFU c. The architecture of GIFU I also contains a part for *nearest power of two* operation, which will be detailed in the following.

The bottom units in Fig. 10 are GIFU c, whose number is equal to the number of cost maps to be filtered. The newest and oldest values of the guidance image are multiplied with the newest and oldest values of the guided cost map, respectively. Next, the multiplied result is input to *mean filter*$_3$ to obtain the correlation result of guided cost map c, i.e., *corr* (I, c). The newest and oldest values of the guided cost map are also input into the *mean filter*$_4$ to obtain *mean* (c). Subsequently, the product of *mean*(c) and *mean*(I) is subtracted from *corr* (I, c) to obtain the covariance of guided cost map c, i.e., *cov* (I, c). *cov* (I, c) is then divided by *var*(I) to obtain a_k. Here, the *nearest power of two* operation is adopted, which uses right shifting to replace the complex division calculation. If *var*(I) is 9, then 8 $(= 2^3)$ is the nearest power of two, and *cov* (I, c) will be shifted right by 3 bits to obtain a_k. After a_k is obtained, the product of a_k and *mean*(I) is subtracted from *mean*(c) to obtain b_k. a_k and b_k are both passed through the mean filter again. Finally, mean(a_k) is multiplied with guidance image I, and *mean*(b_k) is added to obtain the filtered cost maps q.

D. THE WINNER TAKES ALL UNIT

The WTAU is used to select the disparity value of every point with the smallest cost. Fig. 11 shows the hardware architecture of the WTAU for the left image. Its inputs are the 32 filtered cost maps with even disparity values, and its output is the left disparity map. The smallest cost is found through a pairwise comparison. After all of the points are given disparity values with the smallest cost, the left disparity map is generated. The right disparity map is also generated by using the same operation. After the WTAU operation, the left and the right disparity maps are generated at the same time.

E. THE DISPARITY FUSION UNIT

Fig. 12 shows the DFU hardware architecture. The DFU confirms the disparity by using the left-right consistency check. Its inputs are two disparity maps, i.e., a left disparity map and a right disparity map. Its output is one confirmed disparity map. It is worth noting that the parity of every point in the

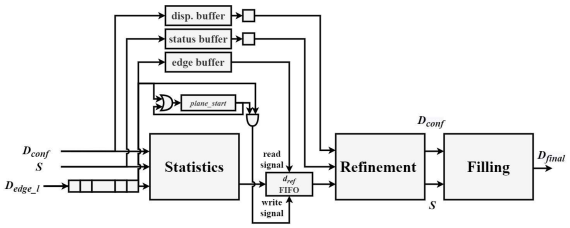

FIGURE 13. The architecture of CPRU.

inputs, i.e., the left disparity map and the right disparity map, is only even or odd, respectively. After the disparity fusion operation, the output, i.e., the confirmed disparity map, can be both. In the proposed DFU architecture, the process of checking consistency is based on the left disparity map, which finds the corresponding point in the right disparity map. Thus, the input mode of the left disparity map is serial-in-serial-out (SISO), and that of the right disparity map is serial-in-parallel-out (SIPO).

Assume that the disparity value at point (i, j) of the left disparity map is k. According to the proposed algorithm, the point (i, j) has to check the consistency with disparity values $k-1$, k, and $k+1$. The first case checks whether the disparity of $D_r(i-(k-1), j)$ is equal to $k-1$. The second case checks the absolute difference between $D_l(i, j)$ and $D_r(i-k, j)$. The third case checks whether the disparity of $D_r(i-(k+1), j)$ is equal to $k+1$. If one of the cases is satisfied, the corresponding valid signal (S_1, S_2, or S_3) is set as 1. According to these three valid signals, the confirmed disparity value, $D_{conf}(i, j)$, can be determined. The earlier confirmed case has a higher priority in the design; that is, the output will be output based on the results obtained for this case. The record position map, S, which represents whether the disparity value at this position is consistent, can also be generated. If one of the cases is fit, which means that the current processing point is consistent, then the value of this point in the position map is set as 1. Otherwise, the value of this point in the position map is set as 0.

F. THE CONTINUOUS PLANE REFINEMENT UNIT

The CPRU is responsible for the refinement of the confirmed disparity map (D_{conf}). As shown in Fig. 13, there are three CPRU inputs: D_{conf}, the left edge map D_{edge_l}, and the position map S. D_{conf} and S are generated by the DFU, and D_{edge_l} is generated from the GCEU.

The CPRU architecture realizes the novel continuous plane refinement method. It comprises three parts: the first is the *Statistics*, which counts the confirmed disparity values between two edges, where the disparity value with highest proportion is found for the purpose of refinement. The second is the *Refinement*, where the highest proportion disparity value is adopted to replace the inconsistent points in the continuous plane. The last is the *Filling*, which assigns the nearest consistent disparity value to those points which are still inconsistent. Since the *Statistics* part of the architecture

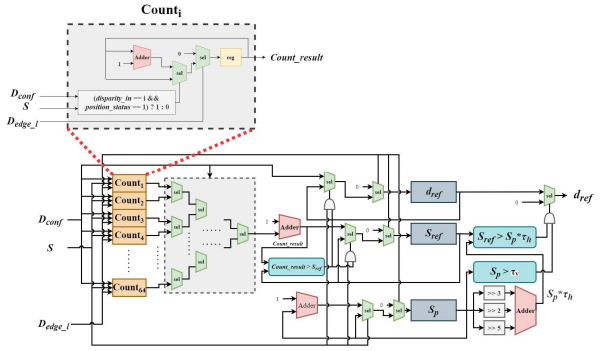

FIGURE 14. The *Statistics* architecture.

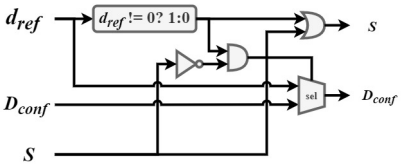

FIGURE 15. The *Refinement* architecture.

needs to count the number of points with each disparity value, three buffers (the disparity buffer, the edge buffer, and the status buffer) are adopted to save the input values. The architecture of each part is detailed in the following.

The *Statistics* architecture is shown in Fig. 14. It contains several counting elements. Each element is used to count the number of consistent points between two edges with one specific disparity value. For example, if the disparity value of the processing point is i, and the value of the processing point in the position map is equal to 1, then the *count result* of i-th counting element will add 1. After the statistical measurements for all of the count elements are completed, the highest proportion disparity value (d_{ref}) is obtained. If the total number of consistent points between two edges (S_p) is greater than the threshold τ_v, and the ratio of the number of points with the highest proportion disparity value to the total number of consistent points (S_{ref}/S_p) is also greater than the threshold τ_h, the d_{ref} is regarded as the disparity value that can be used to refine the inconsistent points in the continuous plane.

The *Refinement* architecture is shown in Fig. 15. It is responsible for replacing the disparity values of the inconsistent points with d_{ref}. For the refinement operation, the consistency of the current input point is checked based on the position map, i.e., S. If the current point is inconsistent, and d_{ref} is valid, the disparity value of the current point is replaced with d_{ref} and its value in the position map is set as 1. By using the refinement process, most of the inconsistent points in the disparity map can be refined effectively. However, in certain cases, the d_{ref} cannot be found; thus, there are still some inconsistent points. These points will be refined with the *Filling* architecture.

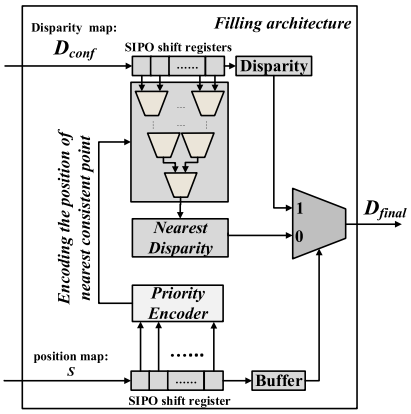

FIGURE 16. The *Filling* architecture.

Fig. 16 shows the *Filling* architecture, which produces the disparity values for points that are still inconsistent after the refinement step. First, the values of the position map are saved in the serial-in/parallel-out (SIPO) shift registers. Subsequently, the *Priority Encoder* uses this information to judge which point is inconsistent. When one inconsistent point is checked, its disparity value is replaced by the disparity value of the nearest consistent point. The remaining inconsistent points are assigned with more reasonable disparity values to improve the overall accuracy with the *Filling* architecture.

G. THE MEDIAN FILTER UNIT

Before outputting, a median filter is applied to the disparity map again. Because the left and the right images are generated by using parallel lenses, there is much less noise in the horizontal direction than in the vertical direction. Therefore, we only perform vertical filtering. The median value is found among $2r + 1$ disparity values, where r (=4) is the radius of the mean filter adopted as discussed in Section V-C. With regard to hardware implementation, 8 BRAMs whose size are equal to the width of the disparity map are adopted to save the disparity values of the previous eight rows. When the disparity value of the ninth row is input, filtering begins. The median finding processing is finished by 19 compare-and-swap circuits [21]. After all disparity values are filtered, the final disparity map is generated.

VI. EXPERIMENTAL RESULTS

This section analyzes the performance of the proposed disparity estimation algorithm, and the algorithm is also compared with state-of-the-art designs in terms of the disparity map error rate, hardware cost, and operating speeds. To validate the performance of the proposed algorithm, it was implemented on Kintex-7 FPGA. Notably, to make a fair comparison of performance with other state-of-the-art works, the Middlebury Version 2 and Version 3 datasets [24] are used as the ground truth in this evaluation. The Version 2 dataset is adopted in [7], [8], [13]–[15], and [19], [20], which contains four sets of ground truths of disparity maps without noise. The Version 3 dataset is adopted in [22], [23], and it provides more complicated scenes with a light-noise environment. The following section provides detailed comparisons made by using these two datasets, and the comparing results between different system configurations are also provided.

A. COMPARISON WITH MIDDLEBURY VERSION 2 DATASET

In addition to our architecture, several state-of-the-art designs [7], [8], [13]–[15], [19], [20], were also included. The earlier designs [7], [8] first implemented disparity estimation methods on FPGA platforms. The later designs [13]–[15] used ADSW-based algorithms. Finally, the latest designs [19], [20], as well as the architecture proposed here used GIF-based disparity estimation algorithms.

Table 1 lists the evaluation results obtained with the Version 2 dataset. Three evaluation items were adopted to analyze with the dataset. The three evaluation items used to evaluate the percentage of bad points were non-occlusion regions (nonocc), all regions (all), and discontinuous regions (disc). The criterion for the percentage of bad points (E_R) is calculated as:

$$E_R = \frac{1}{N_R} \sum\nolimits_{(x,y)\in R} (\left|D_c\,(x,y) - D_{final}\,(x,y)\right| > \tau_d), \quad (14)$$

where R represents the evaluated regions; N_R is the total number of points associated with the evaluated regions; $D_c\,(x,y)$ and $D_{final}\,(x,y)$ are, respectively, the disparity values of the ground truth and our experimental results, and τ_d is a constant threshold. An absolute difference between $D_c\,(x,y)$ and $D_{final}\,(x,y)$ of greater than τ_d, (x,y) represents a bad point. In the evaluation of Table 1, τ_d is set as 1. A disparity value that differs from the ground truth will be considered a bad point. The hardware resource requirements and operating speeds are listed in Table 2. The resource requirement includes the LUT, Slice Register, BRAM, and DSP. The items related to the processing speed are fps (frames per second), MDE/s (million disparity estimations per second) and the operating frequency. The power consumption of the hardware implementation is also listed in Table 2 to examine the efficiency of different designs.

Based on the content of Table 1 and Table 2, the error rates of the ADSW-based designs ([13]–[15]) and GIF-based designs ([19], [20], and the proposed design), were mostly lower than those of the earlier designs ([7], [8]). Furthermore, a comparison of the ADSW- and GIF-based designs indicated that ADSW-based designs have a slightly lower error rate but much higher hardware resource requirements than GIF-based designs. Therefore, in a trade-off between a design's performance and corresponding hardware resource requirements, GIF-based design may be a better solution because they usually provide acceptable performance at a low cost. GIF-based designs are analyzed in detail in the following.

The design proposed in [19] was the first to include a GIF in the disparity estimation algorithm to reduce the area cost under a low error rate; however, its area cost and operating

TABLE 1. Testing performance (error rate) of related studies and proposed architecture with version 2 dataset.

Work	Platform	Tsukuba			Venus			Teddy			Cones			Average Bad Pixel Rate
		nonocc	all	disc	nonocc	all	disc	nonocc	all	disc	nonocc	all	disc	
Jin et. al. 2010 [8]	FPGA	9.79	11.60	20.30	3.59	5.27	36.80	12.50	21.50	30.60	7.34	17.6	21.00	17.24
Zhang et. al. 2011 [7]	FPGA	3.84	4.34	14.20	1.20	1.68	5.62	7.17	12.60	17.40	5.41	11.00	13.90	8.20
Wang et. al. 2015 [13]	FPGA	2.39	3.27	8.87	0.38	0.89	1.92	6.08	12.10	15.40	2.12	7.74	6.19	5.61
Jin et. al. 2014 [14]	FPGA	1.66	2.17	7.64	0.40	0.60	1.95	6.79	12.40	17.10	3.34	8.97	9.62	6.05
Shan et. al. 2014 [15]	FPGA	3.62	4.15	14.00	0.48	0.87	2.79	7.54	14.70	19.40	3.51	11.10	9.64	7.65
Ttofis et. al. 2016 [19]	FPGA	2.38	3.01	9.38	0.40	0.70	3.62	7.23	12.70	17.20	2.87	8.59	8.27	6.36
Vala et. al. 2018 [20]	FPGA	4.50	6.72	11.34	2.30	3.88	9.80	8.61	14.20	18.04	7.28	14.67	15.07	9.73
Proposed	**FPGA**	**2.30**	**3.35**	**10.18**	**0.57**	**1.00**	**5.80**	**5.41**	**10.62**	**14.37**	**4.06**	**9.41**	**9.30**	**6.36**

TABLE 2. Hardware resources required in related studies and proposed architecture (N.M. = not mentioned) with Kintex-7 FPGA.

Work	Image Resolution	Range	fps	MDE/s (10^6)	Freq. (MHz)	LUTs	Slice Registers	BRAMs	DSPs	Power (W)
2010 [8]	640×480	64	230	4522	93	60598	53616	322	12	*N.M.*
2011 [7]	1024×768	64	60	3019	65	53095	74109	*N.M.*	252	*N.M.*
2015 [13]	1024×768	96	68	5120	*N.M*	125255	81092	*N.M.*	*N.M.*	*N.M*
2014 [14]	1024×768	60	199	9404	318	122900	*N.M.*	165	*N.M.*	10.6
2014 [15]	1024×768	128	129	13076	103	60160	33291	*N.M.*	512	*N.M*
2016 [19]	1280×720	64	60	3538	103	57492	71192	302	458	2.8
2018 [20]	1280×720	64	103	6075	107	34181	47368	247	273	2.1
Proposed	**1280×720**	**64**	**118**	**6960**	**110**	**51752**	**35807**	**227**	**258**	**2.6**

speed left room for improvement. The design introduced in [20] was an improvement of that in [19], with the incorporation of a discrete wavelet transform method, where both the left and the right images are zoomed out by a factor of two during the processing stage and finally zoomed in back to the original magnification. This method significantly reduces the area cost but also increases the error rate. By contrast, the design proposed here adopts the novel disparity fusion method, thereby requiring that only half of the disparity values are checked, and the refinement operation CPR is included to enhance the performance.

The average error rate (bad pixel rate) of the proposed algorithm is 6.36%, which is the same as that in [19] and 34.6% lower than that in [20]. With regard to the processing speed, the proposed algorithm can operate at 110 MHz clock frequency and achieve the 118 fps and 6960 MDE/s. These three indicators are better than those in [19] and [20]. In terms of hardware resources, the proposed design requires the fewest Slice Registers, BRAMs, and DSPs. Only the LUT requirement is higher than that in [20]. It is shown that the proposed design can meet the high-speed and low-cost requirements.

B. COMPARISON WITH MIDDLEBURY VERSION 3 DATASET

Similarly, the criterion for the percentage of bad points with the Version 3 dataset (AE_R) was calculated as:

$$AE_R = \frac{1}{N_R} \sum_{(x,y)\in R} d_{err}(x, y) \tag{15}$$

$$d_{err}(x, y) = \begin{cases} 0, & if \ \left|D_c(x, y) - D_{final}(x, y)\right| \le \tau_d \\ \left|D_c(x, y) - D_{final}(x, y)\right|, & otherwise \end{cases} \tag{16}$$

The value of the constant threshold τ_d can be defined by users here. As for the $d_{err}(x, y)$, if the absolute difference between $D_c(x, y)$ and $D_{final}(x, y)$ is greater than τ_d, $d_{err}(x, y)$ is equal to their absolute difference; otherwise, $d_{err}(x, y)$ is 0.

Table 3 lists the evaluation results with the Version 3 dataset. Here, for the purpose of validation, we adopted the designs from [22], [23]. The average error rate of the proposed algorithm was 30.30% when the τ_d was equal to 1.0. Compared to the design from [22], the proposed design led to a 15.11% error rate reduction on average. When the τ_d was equal to 4.0, the error rate of the proposed design is 19.64%. Compared to the design from [23], the proposed algorithm can lead to a 27.98% reduction in the error rate on average. To summarize, the used of the proposed design leads to the lowest error rate when compared with the designs also adopting the Version 3 dataset. As shown in Table 4, in regard to the operating speed and the required hardware resources, the proposed design has the highest MDE/s and operating frequency with competitive hardware resources usage as compared with the designs from [22], [23].

C. COMPARISON BETWEEN DIFFERENT SYSTEM CONFIGURATIONS

In Table 5, the performance of the designs based on the proposed novel disparity fusion method are compared. Middlebury Version 2 dataset is used and the criterion for the percentage of bad pixel points is calculated in the same manner

TABLE 3. Testing performance (error rate) of related studies and proposed architecture with version 3 dataset. (N.M. = not mentioned).

Dataset	*Adir*	*ArtL*	*Jad*	*Mot*	*MotE*	*Pia*	*PiaL*	*Pipe*	*Plr*	*Plt*	*PltP*	*Rec*	*She*	*Ted*	*Vin*
[22] (τ_d =1.0)	23.57%	19.58%	N.M.	17.89%	*N.M.*	26.12%	*N.M.*	17.68%	*N.M.*	45.19%	*N.M.*	21.62%	*N.M.*	10.30%	*N.M.*
Ours (τ_d=1.0)	**23.74%**	**28.29%**	**67.61%**	**9.56%**	**19.12%**	**24.53%**	**49.26%**	**13.29%**	**39.06%**	**32.64%**	**19.54%**	**15.92%**	**43.01%**	**6.21%**	**62.67%**
[23] (τ_d=4.0)	18.60%	22.90%	36.90%	20.60%	18.90%	23.60%	34.00%	23.10%	33.80%	45.80%	22.90%	15.50%	37.30%	12.80%	42.30%
Ours (τ_d=4.0)	**10.98%**	**19.32%**	**61.03%**	**3.82%**	**9.63%**	**14.38%**	**31.04%**	**7.55%**	**22.14%**	**19.88%**	**9.28%**	**5.67%**	**28.66%**	**2.39%**	**48.76%**

TABLE 4. Hardware resources required in related studies and proposed architecture (N.M. = not mentioned) with Kintex-7 FPGA.

Work	Image Resolution	Range	fps	MDE/s (10^6)	Freq. (MHz)	LUTs	Slice Registers	BRAMs	DSPs
2017[22]	640×480	64	324	6370	100	52254	33549	*N.M.*	*N.M.*
2018[23]	1280×720	64	60	3538	78	31135	40187	215	258
Proposed	**1280×720**	**64**	**118**	**6960**	**110**	**51752**	**35807**	**227**	**258**

TABLE 5. Testing performance of the designs based on the proposed disparity fusion method.

Design	Number of Cost Maps	Average Bad Pixel Rate
DF21	42	33.00
DF32	**64**	**22.55**
DF64	128	21.82

TABLE 6. Hardware resources usage of related studies and proposed architecture with different system configurations.

Work	Disp. Range	LUTs	Slice Registers	BRAMs	DSPs	Power (W)
2016 [19]	16	21920	21223	134	122	1.14
	32	33570	27490	190	234	1.64
	64	57492	71192	302	458	2.80
2018 [20]	16	8709	12561	56	68	*N.M.*
	32	18256	24157	120	129	*N.M.*
	64	34181	47368	247	273	2.10
Ours	16	14945	11012	107	66	0.78
	32	27806	19779	147	130	1.41
	64	51752	35807	227	258	2.62

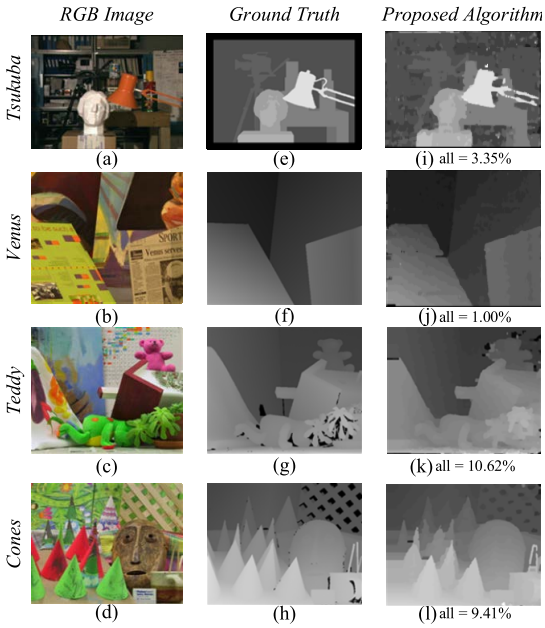

FIGURE 17. The visual testing results for Middlebury Version 2 dataset: (a)–(d) RGB images, (e)–(h) ground truth, (i)–(l) testing results obtained using the proposed algorithm.

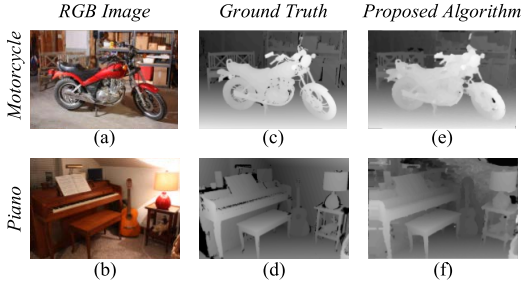

FIGURE 18. The visual testing results for Middlebury Version 3 dataset: (a)–(b) RGB images, (c)–(d) ground truth, (e)–(f) testing results obtained using the proposed algorithm.

as that in Section VI-A. For a fair comparison, no refinement is adopted on the results of Table 5. The design DF21 uses 21 disparity value for the left cost map generation and another 21 disparity values for the right. The design DF32 uses even disparity values for calculating the left cost maps and uses odd disparity values for calculating the right cost maps. The design DF64 uses each disparity value in the disparity range for the left and the right cost maps generation. In the DF21 and DF32, the disparity fusion method has to recover the lost disparity information to generate the disparity map which can cover full disparity range. In the DF64, the disparity fusion method acts just like the general left-right consistency checking method. In the DF21, the left cost maps are calculated with disparity values 3, 6, 9, ... , 60, and 63, and the right cost maps are calculated with disparity values 1, 4, 7, ... , 58, and 61, respectively. If one disparity value from the left disparity map is k, then the disparity fusion method has to check the consistency with disparity values $k-2$, $k-1$, k, and $k+1$. Since the DF21 is not the design adopted for hardware implementation, its checking rules are omitted here for the brevity of the paper. Table 5 shows that

the performance of the DF32 is very close to that of DF64, which implies that the proposed disparity fusion method can effectively recover the lost disparity information. However, the performance of the DF21 is much worse, which resulted from the loss of cost information. As a result, the design DF32 is adopted for hardware implementation for the balance between the performance and calculation complexity.

Table 6 shows the comparison of hardware resources usage between related studies and the proposed architecture under different disparity ranges. It can be observed that the amount of LUTs, slice registers and DSPs of the proposed architecture scales almost linearly. The amount of BRAMs used in the proposed architecture exhibits a quadratic increase, which is proportional to the number of the cost maps.

D. TESTING RESULTS

Fig. 17 and Fig. 18 show the visual testing results. Fig. 17(a)-(d) are the testing images in the Version 2 dataset. Fig. 17(e)-(h) provide the ground truth of the disparity maps, and Fig. 17(i)-(l) show the disparity maps generated by the proposed design. It can be seen that the proposed algorithm could accurately predict most of the disparity values. The bad pixels mostly occurred at the edge of the disparity maps. The visual effects were acceptable. Fig. 18(a)-(b) are the testing images in the Version 3 dataset. Fig. 18(c)-(d) are the ground truth of the disparity maps, and the disparity maps generated by the proposed design are shown in Fig. 18(e)-(f). Fig. 18 shows that the proposed algorithm can still generate high-quality disparity maps even in an environment with light noise.

VII. CONCLUSION

In this study, a high-speed low-cost disparity estimation algorithm was developed and implemented on FPGA platforms. With the proposed disparity fusion method, the number of disparity values to be checked can be reduced. The lost disparity information is recovered by the disparity fusion method. Therefore, the total area cost is reduced significantly and the operating speed is also enhanced. Furthermore, the refinement operation CPR is also included, where the performance of the proposed algorithm was shown to be the same as that obtained by using the disparity algorithms with full disparity range. The experimental results have shown that the proposed architecture can achieved the lowest error rate when compared to other GIF-based designs on Middlebury Version 2 dataset. The tolerance of the proposed architecture to light noise is also verified with Middlebury Version 3 dataset. Moreover, the operating frequency of the proposed architecture is also improved, which allows the frame rate and MDE/s can meet real-time requirements. The experimental results indicated that the design proposed in this work is suitable for applications in resource-constrained end-user equipment because of its high-speed, low-cost property and high-quality results.

REFERENCES

[1] M. Gong and Y.-H. Yang, "Real-time stereo matching using orthogonal reliability-based dynamic programming," *IEEE Trans. Image Process.*, vol. 16, no. 3, pp. 879–884, Mar. 2007.
[2] A. Klaus, M. Sormann, and K. Karner, "Segment-based stereo matching using belief propagation and a self-adapting dissimilarity measure," in *Proc. 18th Int. Conf. Pattern Recognit. (ICPR)*, Aug. 2006, pp. 15–18.
[3] V. Kolmogorov, P. Monasse, P. Tan, and K. Zabih, "Graph cuts stereo matching algorithm," *Image Process. Line*, vol. 4, pp. 220–251, Oct. 2014.
[4] B. Tippetts, D. J. Lee, K. Lillywhite, and J. Archibald, "Review of stereo vision algorithms and their suitability for resource-limited systems," *J. Real-Time Image Process.*, vol. 11, no. 1, pp. 5–25, Jan. 2016.
[5] R. A. Hamzah and H. Ibrahim, "Literature survey on stereo vision disparity map algorithms," *J. Sensors*, vol. 2016, pp. 1–23, Dec. 2015.
[6] R. Kalarot, J. Morris, and G. Gimel'farb, "Performance analysis of multi-resolution symmetric dynamic programming stereo on GPU," in *Proc. 25th Int. Conf. Image Vis. Comput. New Zealand*, Nov. 2010, pp. 1–7.
[7] L. Zhang, K. Zhang, T. S. Chang, G. Lafruit, G. K. Kuzmanov, and D. Verkest, "Real-time high-definition stereo matching on FPGA," in *Proc. 19th ACM/SIGDA Int. Symp. Field Program. Gate Arrays (FPGA)*, 2011, pp. 55–64.
[8] 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 Trans. Circuits Syst. Video Technol.*, vol. 20, no. 1, pp. 15–26, Jan. 2010.
[9] K.-J. Yoon and I. S. Kweon, "Adaptive support-weight approach for correspondence search," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 28, no. 4, pp. 650–656, Apr. 2006.
[10] F. Tombari, S. Mattoccia, and L. D. Stefano, "Segmentation based adaptive support for accurate stereo correspondence," in *Proc. Pacific-Rim Symp. Image Video Technol.*, Jun. 2006, pp. 427–438.
[11] 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. Circuits Syst. Video Technol.*, vol. 20, no. 6, pp. 792–805, Jun. 2010.
[12] J. Ding, J. Liu, W. Zhou, H. Yu, Y. Wang, and X. Gong, "Real-time stereo vision system using adaptive weight cost aggregation approach," *EURASIP J. Image Video Process.*, vol. 2011, no. 1, pp. 1–19, Dec. 2011.
[13] W. Wang, J. Yan, N. Xu, Y. Wang, and F.-H. Hsu, "Real-time high-quality stereo vision system in FPGA," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 25, no. 10, pp. 1696–1708, Oct. 2015.
[14] M. Jin and T. Maruyama, "Fast and accurate stereo vision system on FPGA," *ACM Trans. Reconfigurable Technol. Syst.*, vol. 7, no. 1, pp. 1–24, Feb. 2014.
[15] 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. Embedded Comput. Syst.*, vol. 13, no. 4, pp. 1–24, Jul. 2014.
[16] K. He, J. Sun, and X. Tang, "Guided image filtering," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 35, no. 6, pp. 1397–1409, Jun. 2013.
[17] A. Hosni, M. Bleyer, C. Rhemann, M. Gelautz, and C. Rother, "Real-time local stereo matching using guided image filtering," in *Proc. IEEE Int. Conf. Multimedia Expo*, Jul. 2011, pp. 1–6.
[18] C. Ttofis and T. Theocharides, "High-quality real-time hardware stereo matching based on guided image filtering," in *Proc. Design, Automat. Test Eur. Conf. Exhib. (DATE)*, Mar. 2014, pp. 1–6.
[19] 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 Trans. Comput.*, vol. 65, no. 9, pp. 2678–2693, Sep. 2016.
[20] C. K. Vala, K. Immadisetty, A. Acharyya, C. Leech, V. Balagopal, G. V. Merrett, and B. M. Al-Hashimi, "High-speed low-complexity guided image filtering-based disparity estimation," *IEEE Trans. Circuits Syst. I, Reg. Papers*, vol. 65, no. 2, pp. 606–617, Feb. 2018.
[21] J. Subramaniam, J. K. Raju, and D. Ebenezer, "Fast median-finding word comparator array," *Electron. Lett.*, vol. 53, no. 21, pp. 1402–1404, Dec. 2017.
[22] L. F. S. Cambuim, J. P. F. Barbosa, and E. N. S. Barros, "Hardware module for low-resource and real-time stereo vision engine using semi-global matching approach," in *Proc. 30th Symp. Integr. Circuits Syst. Design Chip Sands (SBCCI)*, 2017, pp. 53–58.

[23] M. Dehnavi and M. Eshghi, "Cost and power efficient FPGA based stereo vision system using directional graph transform," *J. Visual Commun. Image Represent.*, vol. 56, pp. 106–115, Oct. 2018.
[24] *The Middlebury Stereo Vision Page*. Accessed: May 2022. [Online]. Avaliable: http://vision.middlebury.edu/stereo/

YOU-RONG CHEN received the B.S. degree in computer science and information engineering from the National Cheng Kung University, Tainan, Taiwan, in 2020, where he is currently pursuing the Ph.D. degree. His current research interests include image processing, very large-scale integrated chip design, and embedded systems.

WEI-TING CHEN received the B.S. and Ph.D. degrees from the Department of Engineering Science and Computer Science and Information Engineering, National Cheng Kung University, Tainan, Taiwan, in 2017 and 2021, respectively. He is currently a Senior Engineer with Novatek Microelectronics Corporation. His current research interests include image processing, very large-scale integrated chip design, and embedded systems.

SHAO-CHIEH LIAO received the B.S. and M.S. degrees from Kun Shan University, Tainan, Taiwan, in 2010 and 2012, respectively. He is currently pursuing the Ph.D. degree in computer science and information engineering with the National Cheng Kung University, Tainan. His current research interests include circuit design and embedded systems.

PEI-YIN CHEN (Senior Member, IEEE) received the B.S. degree in electrical engineering from the National Cheng Kung University, Tainan, Taiwan, in 1986, the M.S. degree in electrical engineering from Pennsylvania State University, University Park, PA, USA, in 1990, and the Ph.D. degree in electrical engineering from the National Cheng Kung University, in 1999. He is currently a Professor with the Department of Computer Science and Information Engineering, National Cheng Kung University. His research interests include very large scale integration chip design, video compression, fuzzy logic control, and gray prediction.

HONG-YU FANG received the B.S. and M.S. degrees in computer science and information engineering from the National Cheng Kung University, Tainan, Taiwan, in 2017 and 2019, respectively. He is currently a Senior Engineer with Novatek Microelectronics Corporation. His current research interests include image processing, very large-scale integrated chip design, and embedded systems.

TZU-YOU TAI received the B.S. and M.S. degrees in computer science and information engineering from the National Cheng Kung University, Tainan, Taiwan, in 2018 and 2020, respectively. He is currently a Senior Engineer with MediaTek Inc. His current research interests include image processing, very large-scale integrated chip design, and embedded systems.