Received 19 November 2023, accepted 27 November 2023, date of publication 1 December 2023,
date of current version 13 December 2023.

Digital Object Identifier 10.1109/ACCESS.2023.3338649

RESEARCH ARTICLE

Cell-Based Refinement Processor Utilizing Disparity Characteristics of Road Environment for SGM-Based Stereo Vision Systems

CHEOL-HO CHOI, HYUN WOO OH, JOONHWAN HAN, AND JUNGHO SHIN
Pangyo Research and Development Center, Hanwha Systems Company Ltd., Seongnam-si 13524, Republic of Korea
Corresponding author: Cheol-Ho Choi (cheoro1994@hanwha.com)

ABSTRACT Embedded stereo vision systems based on traditional approaches often require a disparity refinement process to enhance image quality. Weighted median filter (WMF)-based processors are commonly employed for their excellent refinement performance. However, when implemented on a field-programmable gate array (FPGA), WMF-based processors face a trade-off between hardware resource utilization and refinement performance. To address this trade-off, we previously proposed a new disparity refinement processor based on the hybrid max-median filter (HMMF). However, our earlier work did not guarantee flawless operation in large occluded and texture-less regions, particularly in areas with numerous holes. In order to overcome this limitation of conventional processors, we proposed a cell-based disparity refinement processor. This processor extends our previous HMMF-based disparity refinement processor. To evaluate its refinement performance, we conducted experiments using four types of publicly available stereo datasets. When comparing refinement performance, our proposed processor outperforms conventional processors when using the KITTI 2012 and 2015 stereo benchmark datasets. Additionally, the results demonstrate that our proposed processor exhibits superior refinement performance when applied to the Cityscapes and StereoDriving datasets in comparison to conventional processors. Furthermore, when considering hardware resource utilization, our proposed processor demonstrates lower resource requirements than conventional processors when implemented on an FPGA. Therefore, our proposed disparity refinement processor is well-suited for the disparity refinement process in stereo vision systems that require cost-effectiveness and high performance.

INDEX TERMS Stereo vision, disparity refinement, hardware architecture, FPGA.

I. INTRODUCTION

As the automotive vehicle market continues to expand, there is a growing focus on advanced driver assistance systems (ADAS) [1], [2], [3]. Among the components of ADAS, the Forward Collision Avoidance (FCA) function plays a pivotal role in preventing accidents involving vehicles and passengers, thereby ensuring the safety of drivers and passengers [4], [5]. To enable the FCA function, acquiring three-dimensional (3D) distance information becomes essential [6]. This crucial information is typically derived from a disparity map, which can be readily converted into a depth map [7], [8]. On embedded platforms, the computation of the disparity map often relies on various traditional stereo matching algorithms, which can be broadly categorized into three main groups: 1) local matching (LM), 2) global matching (GM), and 3) semi-global matching (SGM) [9], [10], [11].

LM generates the disparity map by computing the correlation. The lowest matching cost value is selected using the winner-takes-all (WTA) strategy within a predefined area [12]. LM is often referred to as window-based stereo matching due to its window-based operation [13]. This means that LM can be efficiently implemented using a pipeline architecture on a field-programmable gate array

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

(FPGA). Consequently, it ensures real-time performance on FPGA platforms due to its low computational complexity and operational simplicity. Therefore, it is well-suited for resource-limited embedded real-time stereo vision systems. However, in texture-less or occluded regions, mismatching values may occur because the result is computed based on the correlation of edge information, potentially leading to reduced matching accuracy in these areas [14].

To improve matching accuracy, especially in texture-less and occluded regions, GM (e.g., graph cut, belief propagation, and dynamic programming) is employed [15], [16], [17], [18]. GM computes the disparity value using a global energy function, followed by an energy minimization process based on the Markov random field [19], [20]. Therefore, GM is often referred to as energy-based stereo matching. However, GM requires a significant amount of processing time because it necessitates the use of the entire left and right-side input images. Consequently, when implemented on an FPGA, a memory-based frame grabber module is essential to retain the input left and right-side stereo images. Additionally, the algorithm for the energy minimization process is more complex than that of LM. Hence, a high-performance environment, including a central processing unit (CPU) or a graphics processing unit (GPU), is more suitable than a resource-limited embedded platform when using GM.

SGM combines the concepts of GM and LM to achieve real-time operation on embedded platforms, offering high matching accuracy with reasonable hardware resource utilization, especially when implemented on an FPGA [15], [21]. SGM can compute a dense and smooth disparity map in real-time operation. However, in occluded regions where significant object information is hidden, "holes"—mismatching values that degrade matching accuracy—can occur. This is because SGM utilizes the LM (Local Matching) concept. Therefore, a disparity refinement process is required in the post-processing step to compute the refined disparity map, which exhibits improved matching accuracy compared to the initial disparity map obtained solely from SGM [22], [23].

In disparity refinement methods, the two-dimensional (2D) weighted median filter (WMF) is widely used due to its demonstrated high refinement performance [24], [25], [26]. However, the drawback of the 2D WMF is its substantial demand for hardware resource utilization when implemented on an FPGA, primarily because the 2D WMF has a computational complexity of $O(r^2)$, indicating that its processing demands increase quadratically with the filter radius r [24]. Recent research aims to alleviate hardware resource requirements by reducing computational complexity. To achieve this, Chen et al. proposed a separable WMF (sWMF), and subsequently, Hyun et al. proposed a sparse-window-approach-based sWMF (ssWMF) to further decrease hardware resource utilization compared to the sWMF [27], [28]. While these subsequent studies succeed in reducing hardware resource utilization for implementation on an FPGA, they often come at the cost of degraded refinement performance, revealing a trade-off between hardware resource utilization and refinement performance.

To address these challenges, we conducted an extensive analysis of the disparity characteristics in both the horizontal and vertical directions, leading to the development of an improved disparity refinement architecture that leverages the observed disparity tendency [29]. Our previously proposed disparity refinement architecture exhibited high refinement performance while requiring fewer hardware resources than WMF-based architectures. However, we identified a drawback: the refinement performance deteriorate when holes are abnormally clustered. To overcome this limitation, in the present study, we introduce a cell-based disparity refinement architecture, an extension of our prior work.

II. PREVIOUS WORK

A. TWO-DIMENSIONAL WEIGHTED MEDIAN FILTER

2D WMF generally used bilateral weight for sorting process to select the median value; and bilateral weight is computed by using (1).

$$w(x, x') = exp(\frac{-|I(x) - I(x')|^2}{\sigma^2}). \tag{1}$$

where, $I(x)$ represents the center pixel value in the window of the input image; $I(x')$ denotes the neighboring pixel value based on the center pixel; σ is the value used to measure color similarity; and $w(x, x')$ is the bilateral weight value used for selecting the median value, and it is computed based on the color similarity and the spatial distance between pixels. The 2D WMF has the advantage of preserving edge information and has demonstrated high refinement performance in previous studies. However, when implemented on an FPGA, it poses challenges due to its significant hardware resource requirements. The primary reason for the substantial resource utilization of the 2D WMF is its computational complexity, which scales with the square of the window radius r, where 'r' is the radius of the window. Additionally, the dual-port Block Random Access Memory (BRAM) is required to store the bilateral weights for the sorting process. In other words, when the window size is N the 2D WMF needs to process and store N^2 weight and pixel values for the sorting process. Given these resource demands, the 2D WMF may not be suitable for stereo vision systems that prioritize low-cost characteristics.

B. SEPARABLE WEIGHTED MEDIAN FILTER

Chen et al. introduced the sWMF architecture as a means to reduce the computational complexity of the 2D WMF while maintaining a low-cost characteristic. The sWMF achieves this by incorporating the concept of separable operations, which involves decomposing the filtering process into two separate 1D operations: one in the horizontal direction and one in the vertical direction. By utilizing this separable operation concept, the computational complexity of the sWMF is significantly reduced from $O(r^2)$ to $O(r)$, where r represents the size of the window. Consequently, the sWMF architecture demands fewer hardware resources when implemented on an FPGA, making it a more resource-efficient choice compared to the 2D WMF architecture. Furthermore, the reduced computational complexity of the sWMF has a positive impact on its latency. The sWMF architecture exhibits lower latency than the 2D WMF architecture due to the reduced computational workload, making it a suitable option for applications where real-time or low-latency processing is essential. In summary, the sWMF architecture presents an efficient approach to reduce both computational complexity and hardware resource utilization compared to the 2D WMF architecture, making it an attractive option for various low-cost and real-time processing scenarios.

C. SPARSE-WINDOW-APPROACH-BASED SEPARABLE WEIGHTED MEDIAN FILTER

Indeed, the sWMF architecture has the advantage of reducing hardware resource utilization, but it still requires a significant amount of hardware resources when the window size becomes larger. To address this concern and further optimize hardware resource utilization, Hyun et al.proposed the ssWMF architecture. The ssWMF architecture introduced a sparse-window approach, which involves utilizing fewer pixel values compared to conventional WMF-based architectures. By doing so, this architecture can effectively reduce the hardware resource requirements when implemented on an FPGA. However, it is essential to note that this reduction in hardware resources comes at a cost – the refinement performance of the ssWMF architecture may be compromised compared to conventional WMF-based architectures. The key trade-off with the sparse-window approach is that, because fewer pixel values are used for sorting and selecting processes, there can be a reduction in refinement performance. This is especially noticeable when the window is positioned over a clustered hole region. In such cases, the ssWMF architecture may encounter critical issues that impact its ability to produce accurate results. In essence, the ssWMF architecture offers a resource-efficient alternative but may not perform as well in scenarios where disparity refinement relies heavily on the information within the window, particularly when dealing with clustered hole regions.

D. OUR PREVIOUS WORK

In studies involving methods based on WMF, they showed trade-off between hardware resource utilization and the effectiveness of disparity refinement. When weight-based image processing is implemented using hardware on an FPGA, there is a concern that it may consume a significant amount of hardware resources, primarily due to the utilization of dual-port BRAM and other factors. To address this concern, as depicted in Fig. 1, we conducted a comprehensive analysis of the disparity characteristics in road environments, both in the horizontal and vertical directions [29]. In the vertical direction, we observed a gradual increase in disparity values from the top coordinate to the bottom coordinate. This gradual disparity increase signifies a corresponding depth value increase from the bottom to the top coordinate [30]. Conversely, in the horizontal direction, we discovered that preserving edge information is crucial for enhancing refinement performance. Building upon these analysis results, we introduced the Hybrid Max-Median Filter (HMMF) architecture. In our previously proposed architecture, the result value of the refined disparity map is computed using eight inner sub-windows that cover eight distinct directions, including horizontal, vertical, and diagonal orientations. The result value is determined by initially applying the max filter and then the median filter. This architectural approach excels in accurately capturing the disparity characteristics. However, it is essential to note that, similar to the ssWMF architecture, the refinement performance of our architecture may be compromised when the window is positioned over a clustered hole region. In such cases, the reduced number of pixel values used for the selection process can lead to performance degradation.

III. PROPOSED METHOD

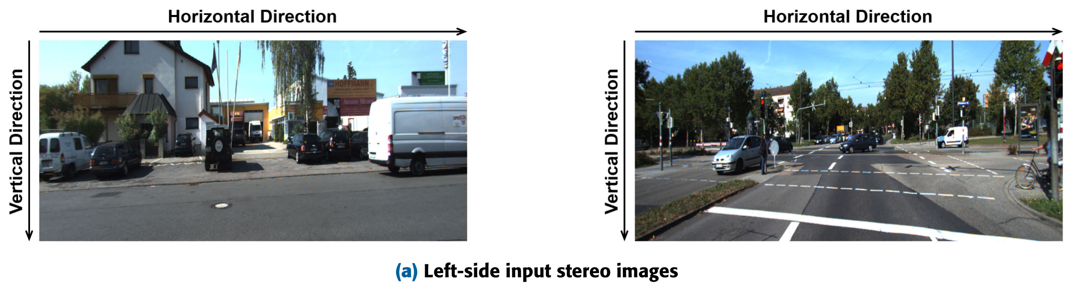
To address the limitation identified in the HMMF method from our previous study, we have introduced a cell-based disparity refinement method. Fig. 2 provides an overview of the operational process of our proposed method in the current study. As illustrated in Fig. 2, our proposed method consists of two key steps: 1) Cell and Inner-window Generation and 2) Disparity Characteristics-Reflecting Computation.

A. CELL AND INNER-WINDOW GENERATION

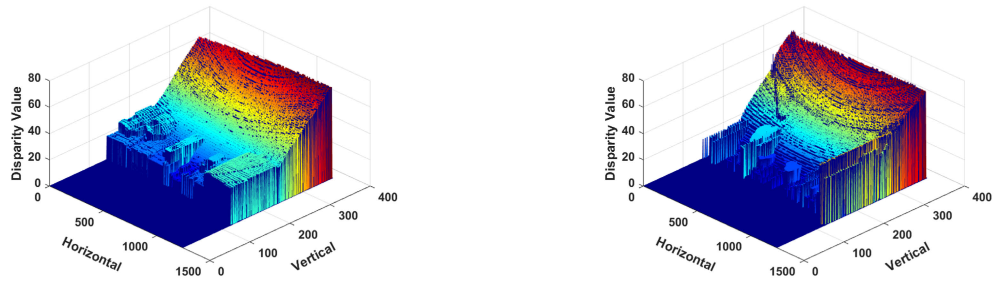
This step involves the creation of cells and inner-windows, which serve as the fundamental units for disparity refinement In Fig. 2, four cells and four inner-windows are generated using (2)–(12).

$$a = (w+1)/2 \tag{2}$$
$$b = (h+1)/2 \tag{3}$$
$$Cell_1 = I[1:a-1][1:b-1] \tag{4}$$
$$Cell_2 = I[1:a-1][b+1:h] \tag{5}$$
$$Cell_3 = I[a+1:w][1:b-1] \tag{6}$$
$$Cell_4 = I[a+1:w][b+1:h] \tag{7}$$
$$Inner_1 = I[a][1:b] \tag{8}$$
$$Inner_2 = I[a:w][b] \tag{9}$$
$$Inner_3 = I[a][b:h] \tag{10}$$
$$Inner_4 = I[1:a][b] \tag{11}$$
$$W_C = I[a][b] \tag{12}$$

where w and h represent the width and height sizes of the predefined window; a and b denote the coordinates for the horizontal and vertical directions where the center pixel is located within the predefined window; W_C represents the center pixel value; $Cell_1$, $Cell_2$, $Cell_3$, and $Cell_4$ correspond to the generated cells for the north-west, south-west, north-east,



(a) Left-side input stereo images



(b) Ground truth of the corresponded left-side input images

FIGURE 1. Left-side input stereo image and three dimensional (3D) plot for analyzing the disparity tendency in the road environment-based disparity map.

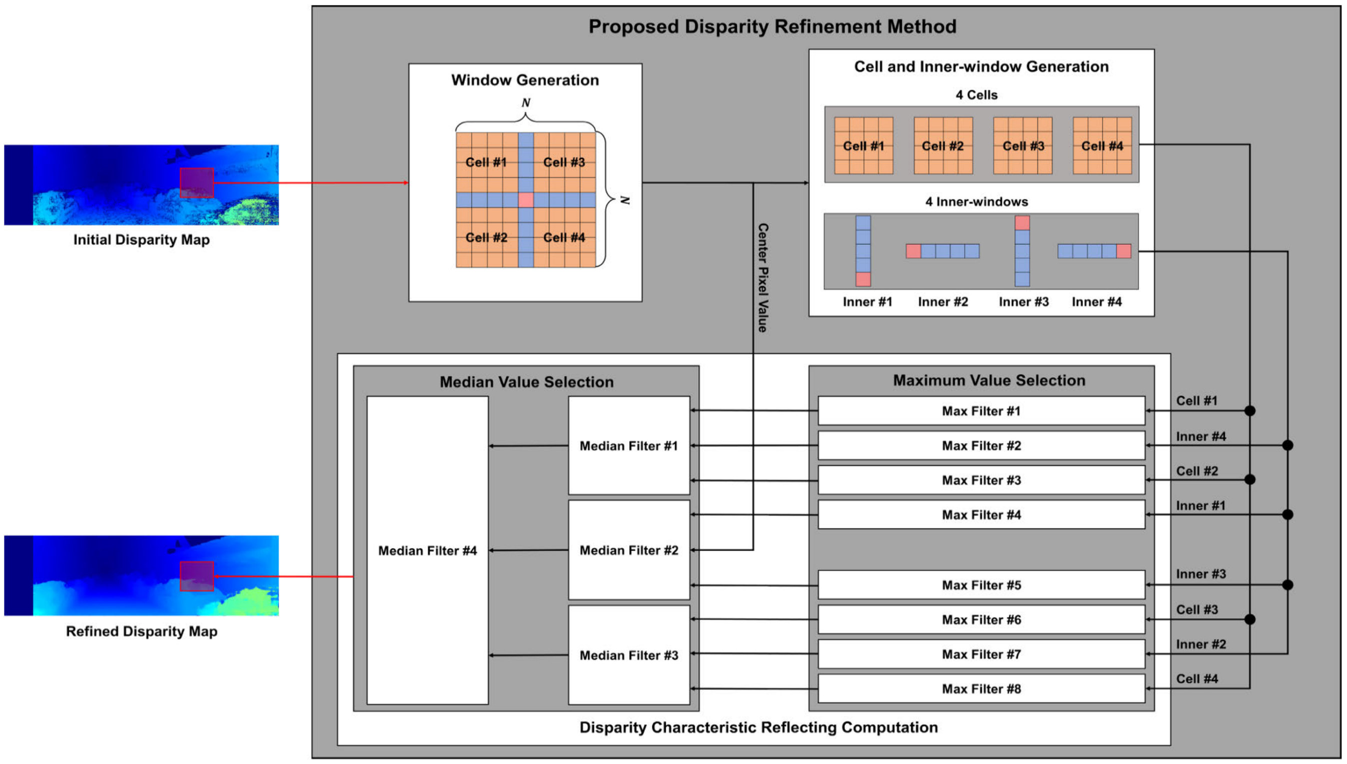


FIGURE 2. Operation process of the proposed disparity refinement method.

and south-east directions, respectively; and $Inner_1$, $Inner_2$, $Inner_3$, and $Inner_4$ refer to the generated inner-windows for the upper vertical, right horizontal, lower vertical, and left horizontal directions, respectively.

Regarding the inner-windows, two are generated for the left and right-side horizontal directions based on the center pixel, while the other two are generated for the upper and lower-side vertical directions based on the center pixel. When using a 13×13 window size, as shown in (11), the left horizontal inner-window is composed of seven pixel values located from $(1, 7)$ coordinate to $(7, 7)$ coordinate. As shown in (9), the seven pixel values from $(7, 7)$ to $(13, 7)$ coordinates are used for generating the right horizontal inner-window. On the vertical direction, the pixel values from

(7, 1) coordinate to (7, 13) coordinate are used for generating two vertical inner-windows. In terms of the upper direction, as shown in (8), the seven pixel values from (7, 1) coordinate to (7, 7) coordinate are used for generating inner-window. In terms of the lower direction, as shown in (10), the seven pixel values from (7, 7) coordinate to (7, 13) coordinate are used for generating inner-window. As a result, each inner-window must include the center pixel to ensure an odd number of pixels and reflect the center pixel value, as required for our proposed method.

Regarding the cells, the four cells are generated using the pixel values, excluding pixels used for inner-window generation. Therefore, each cell has a square shape when using the square input window size. In Fig. 2, cell numbers one and two are generated for the left-side diagonal direction, while cell numbers three and four are generated for the right-side diagonal direction. When using a 13 × 13 window size, the number one cell is generated using thirty-six pixel values from (1, 1) coordinate to (6, 6) coordinate, as shown in (4). In the number two cell, it is generated by using thirty-six pixel values from (1, 8) coordinate to (6, 13) coordinate, as shown in (5). In the number three cell, it is generated by using thirty-six pixel values from (8, 1) coordinate to (13, 6) coordinate, as shown in (6). In the number four cell, it is generated by using thirty-six pixel values from (8, 8) coordinate to (13, 13) coordinate, as shown in (7). The reason for generating four cells is to analyze the disparity values for each diagonal direction, including north-west, north-east, south-west, and south-east.

B. DISPARITY CHARACTERISTICS-REFLECTING COMPUTATION

After the cell generation steps, we employ a characteristic-reflecting computation process with the aim of computing refined pixel values that reflect the disparity characteristics in both the vertical and horizontal directions. In our previous work, we observed distinct characteristics in the vertical and horizontal directions of the disparity map. In the vertical direction, we noted that the disparity values gradually increase from the top coordinate to the bottom coordinate. This observation aligns with the natural progression of depth in typical scenes. Conversely, in the horizontal direction, we found that edge information must be well preserved. This preservation contributes significantly to improved matching accuracy, especially in occluded and texture-less regions. To leverage these observed characteristics, the characteristic-reflecting computation process involves two operation steps: 1) maximum value selection and 2) median value selection.

Equations (13)–(18) encompass the mathematical expressions for the computation of disparity characteristics, which include the steps of maximum value selection and median value selection.

$$CellM_i = max(Cell_i) \tag{13}$$

$$InnerM_i = max(Inner_i) \tag{14}$$

$$M_1 = median(CellM_1, Inner_4, CellM_2), \tag{15}$$

$$M_2 = median(InnerM_1, W_C, InnerM_3), \tag{16}$$

$$M_3 = median(CellM_3, InnerM_2, CellM_4), \tag{17}$$

$$Out = median(M_1, M_2, M_3), \tag{18}$$

where $CellM_i$ represents each selected maximum value of each cell for i numbers ranging from 1 to 4; $InnerM_i$ represents each selected maximum value of each inner-window for i numbers ranging from 1 to 4; W_C represents the center pixel of the input window; M_1, M_2, and M_3 represent vertical median values; and Out represents the final selected median value, which is also the output value for the refined disparity map. In the maximum value selection step, our objective is to identify the eight maximum values from the pre-generated four cells and four inner-windows using (13)–(14). This step is designed to capture the vertical disparity tendency, where disparity values gradually increase. Following this, the median value selection step consists of two phases. In the first phase, we perform the median value selection operation, as illustrated in Fig. 2, using (15)–(17). The selected median value offers the advantage of effectively preserving edges when a median filter is applied. To select this median value, we consider nine pixel values, which include the eight maximum values and the center pixel. The inclusion of the center pixel is crucial because it allows the determination of the median value when the number of input pixels is odd. Additionally, in cases where the center pixel is located at the edges of objects, its value is considered when selecting the three median values. Subsequently, we perform an additional median value selection operation to calculate the output value of the refined disparity map, as depicted

Algorithm 1 Proposed Disparity Refinement Pseudo Code

```
Input: $C, C_1, C_2, C_3, C_4, I_1, I_2, I_3, I_4$
    $C$: Center pixel value
    $C_1$: Cell for north-west direction
    $C_2$: Cell for south-west direction
    $C_3$: Cell for north-east direction
    $C_4$: Cell for south-east direction
    $I_1$: Inner window for upper vertical direction
    $I_2$: Inner window for right horizontal direction
    $I_3$: Inner window for lower vertical direction
    $I_4$: Inner window for left horizontal direction
Output: $O$: Output image
 1: for $i \leftarrow 1$ to 9 do
 2:     if $i \leq 4$ then
 3:         $V_{max}[i] \leftarrow MaximumSelection(C_i)$
 4:     else if $i = 5$ then
 5:         $V_{max}[i] \leftarrow C$
 6:     else
 7:         $V_{max}[i] \leftarrow MaximumSelection(I_{i-5})$
 8:     end if
 9: end for
10: $O \leftarrow MedianSelection(V_{max})$
```

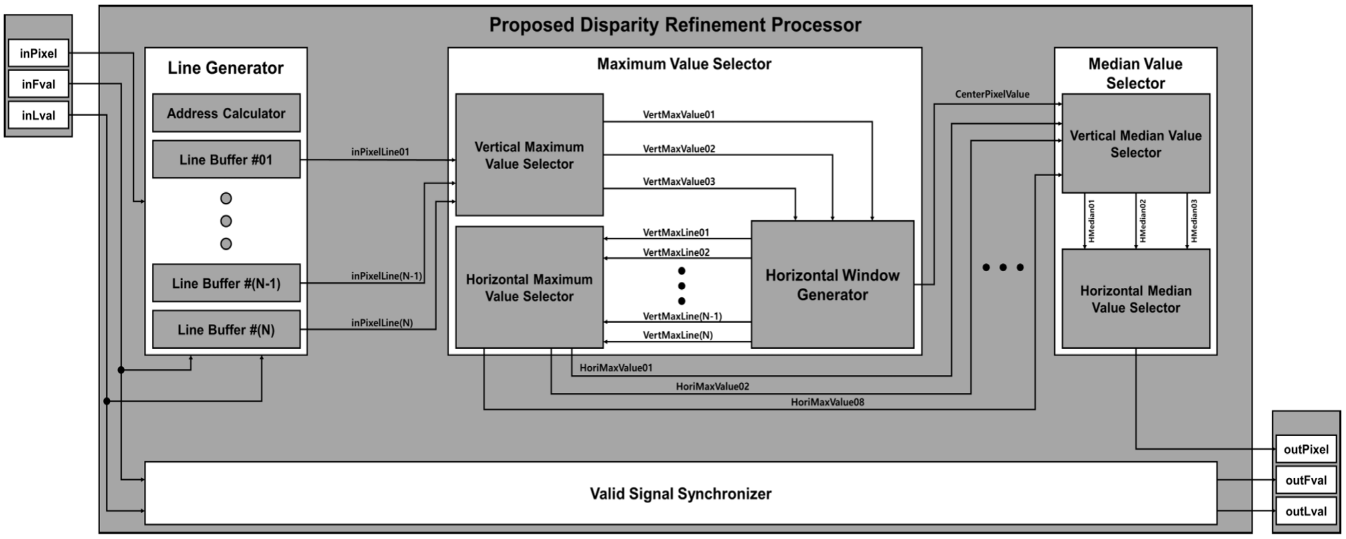


FIGURE 3. Top architecture of the proposed disparity refinement processor.

in Fig. 2. In essence, (15)–(18) enables the preservation of edge information in the horizontal direction. In summary, the characteristic-reflecting computation operation computes the refined pixel value by considering both vertical and horizontal disparity characteristics. This results in a disparity map that effectively reflects these characteristics.

IV. PROPOSED HARDWARE ARCHITECTURE

To implement our proposed cell-based disparity refinement method on the FPGA platform, we proposed a hardware architecture, as depicted in Fig.3. Fig.3 presents the proposed hardware architecture from a top-level module perspective, comprising five sub-modules: 1) Line Generator, 2) Maximum Value Selector, 3) Median Value Selector, and 4) Valid Signal Synchronizer.

A. LINE GENERATOR

The primary objective of the Line Generator module is to create N lines for an input window with an $N \times N$ window size. The Line Generator module comprises the Address Calculator sub-module and N line buffers.

Within the Address Calculator sub-module, it calculates the frame and line valid counting values to generate the write enable signal for BRAM. The line valid signal represents the synchronization signal for the horizontal direction, often referred to as the Hsync signal. Meanwhile, the frame valid signal serves as the synchronization signal for the vertical direction, commonly known as the Vsync signal. For instance, when operating at a resolution of 1280×720, commonly known as high definition (HD) resolution, the line valid counting values are calculated based on the input frame and line valid signals.

For instance, when operating at a resolution of 1280×720, the timing diagram of the valid counter is illustrated in Fig.4. As shown in the timing diagram, the line valid counting values are calculated based on the input frame and line valid signals. This implies that the line valid counting value increases only when both the input frame and line valid signals have a digital value of one. Consequently, when using HD resolution, the line valid counting value is initialized to zero when it reaches 1279 or when the input line valid signal is zero.

Regarding the frame valid counting value, it is calculated based on the input frame valid signal and the line valid counting value. As a result, when the frame valid signal has a digital value of one and the line valid counting value reaches 1279, the frame valid signal is incremented. Therefore, when employing HD resolution, the frame valid counting value is initialized to zero when the input frame valid signal is zero or when both the frame and line valid counting values reach 719 and 1279, respectively. Based on the calculated frame and line valid counting values, the write enable signal is generated for each line buffer.

The N line buffers are designed using BRAM intellectual property (IP). By utilizing BRAM, a vertical window can be generated because input pixel values are stored step by step according to the write enable signal. In other words, generating a vertical window means that N lines required for an operation based on the $N \times N$ window can be created. Additionally, by employing BRAM for the line buffers, not only can a upper-side zero-padding effect be achieved, but a sliding window operation can also be performed.

B. MAXIMUM VALUE SELECTOR

As shown in Fig. 3, N pixel values generated by the Line Generator module, ranging from $inPixelLine01$ to $inPixelLine(N)$, are input into the Maximum Value Selector module to determine the maximum values for each cell and inner-window area. The Maximum Value Selector module comprises three sub-modules: 1) Vertical Maximum Value Selector, 2) Horizontal Window Generator, and 3) Horizontal Maximum Value Selector.

Vertical Maximum Value Selector sub-module comprises two max filters and one center pixel synchronizer. In the

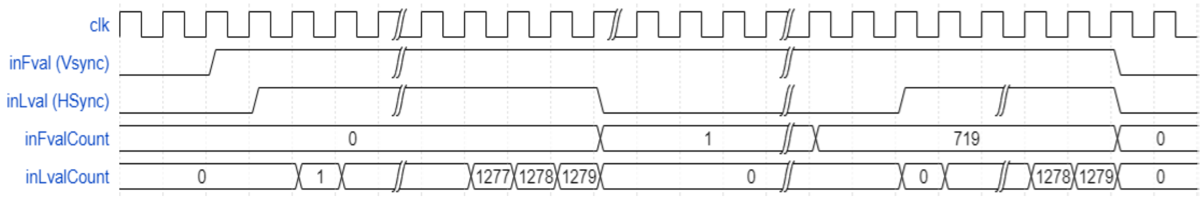


FIGURE 4. **Timing diagram for address calculator in the line generator sub-module.**

Vertical Maximum Value Selector, two max filters, utilizing a pyramidal architecture with comparators, select and output the maximum values ($VertMaxValue01$ and $VertMaxValue03$) for the top and bottom directions. When using a 13×13 window size, as shown in Fig. 5, the max filter requires 4 clock cycles to select the maximum values for the top and bottom directions. Therefore, a Center Pixel Synchronizer based on registers is used to synchronize the center pixel value with the 4-clock maximum value selection operation, resulting in the calculation of output values for the center line ($VertMaxValue02$).

Following the Maximum Value Selector, these three calculated values are input into the Horizontal Window Generator sub-module, which serves two primary functions. Firstly, it generates a horizontal window with a width determined by the window size, utilizing registers exclusively for this purpose. For instance, when employing a 13×13 window size, it necessitates 13 registers for each input value, including $VertMaxValue01$, $VertMaxValue02$, and $VertMaxValue03$. Consequently, when using a 13×13 window size, this sub-module requires 39 registers for horizontal window generation. In general, for an $N \times N$ window size, this sub-module requires $3N$ registers.

Secondly, it selects synchronized center pixel values in coordination with the Horizontal Maximum Value Selector sub-module. This synchronization is achieved by introducing a delay in the center pixel value using registers, corresponding to the latency of the Horizontal Maximum Value Selector sub-module.

After the data passes through the Horizontal Window Generator, the pixel values of the horizontal window are then input into the Horizontal Maximum Value Selection sub-module. This primary aim of the sub-module is to select the eight maximum values, distributed among four cells and four inner-window areas. To accomplish this, the sub-module consists of six max filters and four comparators when employing a square window size. The six max filters are dedicated to processing the four cells and the horizontal inner-window areas in both left and right directions. For the vertical inner-window in the top and bottom directions, max filters are unnecessary, as each maximum value is already selected in the Vertical Maximum Value Selector. However, each selected maximum value for an inner-window necessitates one additional comparator in comparison to the maximum values of cell areas. This additional comparator is crucial because the selected maximum values, identified using max filters for inner-window areas, are not directly compared to the center pixel. Consequently, four comparators are employed to determine the ultimate maximum value for the four inner-window areas.

C. MEDIAN VALUE SELECTOR

After the Maximum Value Selector, the module receives the nine selected pixel values, which are then directed to the Median Value Selector module. The primary objective of this module is to designate the median value as the output for the refined disparity map. In order to optimize the utilization of hardware resources, we've implemented a separable operation for both the horizontal and vertical directions within the Median Value Selector module. This approach has led to the module's division into two distinct sub-modules: 1) Horizontal Median Value Selector and 2) Vertical Median Value Selector.

The Horizontal Median Value Selector sub-module is equipped with three median filters. This configuration remains consistent, even when dealing with a larger window size, as only nine pixel values, comprising eight maximum values and the center pixel, are provided as input. Each median filter employs three pixel values to determine the horizontal median value such as (13)–(17). These median filters have been meticulously designed with a step-by-step operational architecture employing comparators, necessitating three clock cycles to select the output value.

In contrast, within the Vertical Median Value Selector sub-module, a single median filter is utilized. Therefore, in the Horizontal Median Value Selector sub-module, the same max filter architecture is employed to determine the ultimate median value for the output value of the refined disparity map. Consequently, due to this design choice, the Median Value Selector module necessitates a total of six clock cycles to determine the output value for the refined disparity map.

D. VALID SIGNAL SYNCHRONIZER

The Valid Signal Synchronizer module is designed with the primary objective of generating the output valid signal. To accomplish this task, the module incorporates registers that are used to delay the input frame and line valid signals, as shown in the Center Pixel Synchronizer sub-module in Fig.5. For example, when utilizing a 13×13 window size, it becomes necessary to introduce a delay of 29 operation clocks to the input frame and line valid signals. This is due to the operational requirements of other modules, such as the Line buffer, Maximum Value Selector, and Median Value Selector, which demand 2, 21, and 6 operation

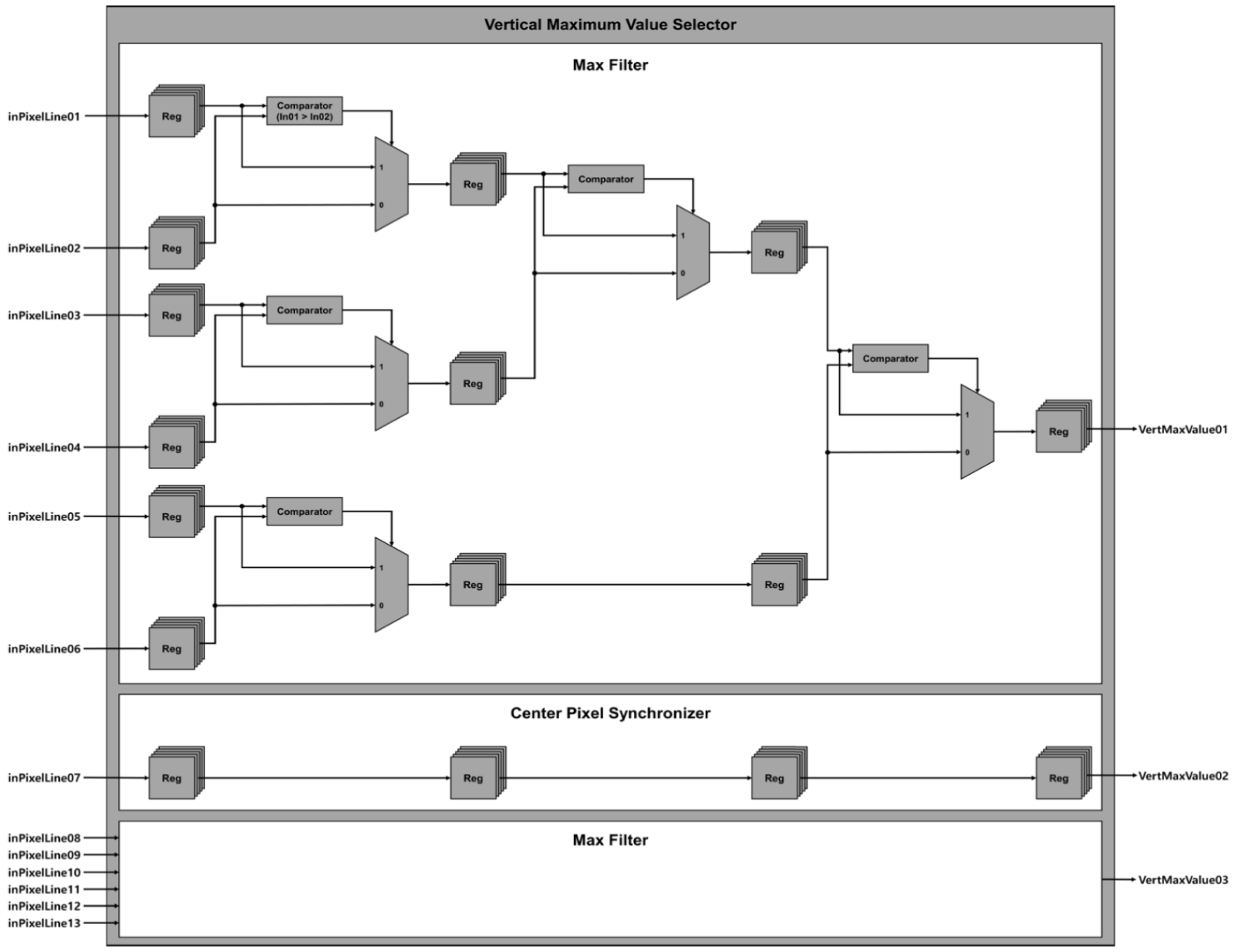


FIGURE 5. **Schematic for vertical maximum value selector module consisting of two max filters and one center pixel synchronizer.**

clocks, respectively. In summary, within the Valid Signal Synchronizer module, the output frame and line valid signals are generated based on the total number of operation clocks required by the proposed disparity refinement processor to compute the output value of the refined disparity map.

V. EXPERIMENTAL RESULTS

To compare the performance of the proposed disparity refinement processor, we conducted two types of experiment categories: 1) refinement performance comparison and 2) hardware performance comparison.

A. REFINEMENT PERFORMANCE COMPARISON

To perform a comprehensive comparison of the refinement performance between the proposed and conventional disparity refinement processors, we conducted experiments using three publicly available stereo datasets: 1) KITTI 2012/2015 stereo benchmark dataset [31], [32], 2) DrivingStereo dataset [33], and 3) Cityscapes dataset [34]. To maintain the integrity and fairness of our experiments, we adhered to the same conditions as those established in previous studies. To achieve this consistency, we leveraged the disparitySGM built-in function provided by MATLAB. This function necessitates the specification of two input parameters: 1) DisparityRange and 2) UniquenessThreshold. DisparityRange parameter defines the range of disparity values considered during the computation. UniquenessThreshold sets the minimum uniqueness value required for a disparity value to be deemed valid. To ensure a fair and meaningful comparison with prior research, we set the UniquenessThreshold to 5 and configured the DisparityRange as [0 128]. Furthermore, for the WMF-based processors, we maintained the color similarity parameter at a constant value of 3. These specific parameter settings guarantee that our experiments are conducted under uniform conditions and enable us to evaluate the proposed disparity refinement processor against established benchmarks and previous research in a robust and equitable manner.

1) KITTI STEREO BENCHMARK

Fig. 6 presents experimental results for visually confirming the initial disparity map and refined disparity maps using

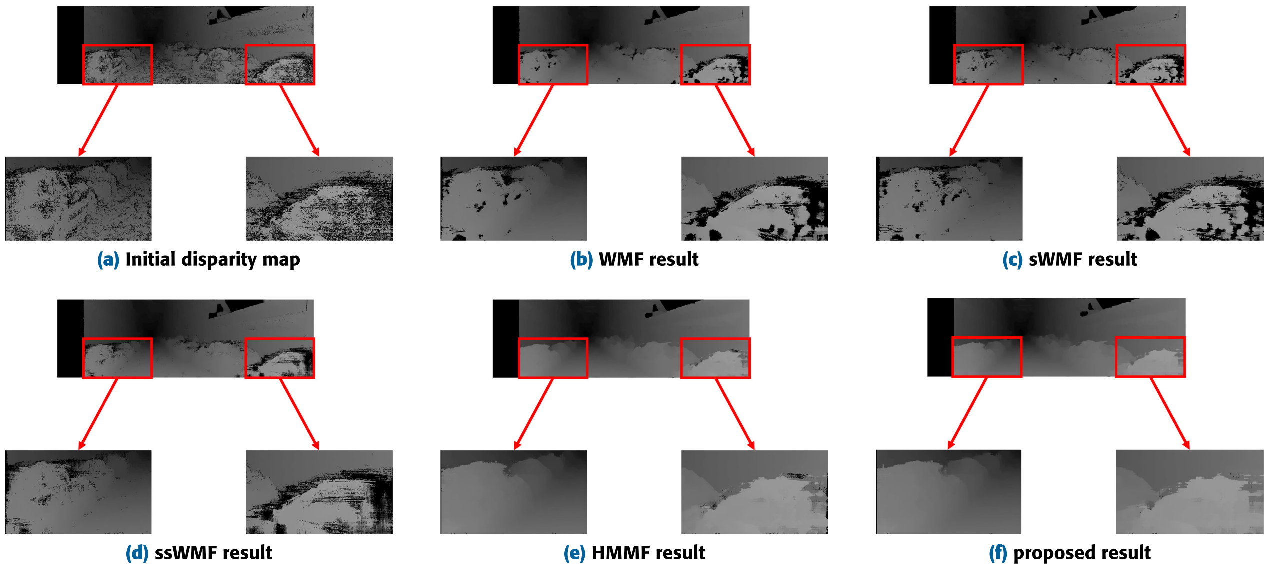


FIGURE 6. Experimental results for visual confirmation on KITTI 2012 stereo benchmark dataset.

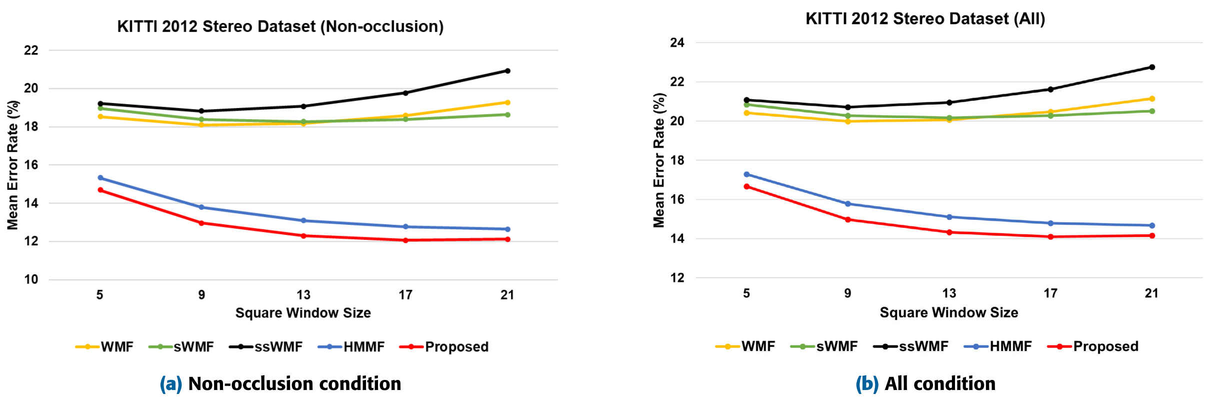


FIGURE 7. Mean error rate (MER) performance of the proposed and conventional disparity refinement processors when using the KITTI 2012 stereo benchmark dataset.

both the proposed and conventional disparity refinement processors. As seen in Fig.6.(a), the initial disparity map generated using SGM reveals a substantial number of hole regions situated along the edges of vehicles. This observation underscores the critical necessity of the disparity refinement process since a significant number of hole regions persist even upon visual examination. In Fig.6.(c)–(d), which illustrate the outcomes of employing WMF-based disparity refinement processors, a noticeable reduction in hole regions is evident compared to the initial disparity map. Nevertheless, in edge regions, it becomes apparent that inaccurate values within the hole regions are not effectively replaced with accurate ones.

In contrast, Fig.6.(e)–(f) demonstrate the performance of HMMF-based and proposed disparity refinement processors. These processors incorporate computations that take into account the disparity trends in both the vertical and horizontal directions, resulting in superior refinement performance when contrasted with WMF-based disparity refinement processors. However, as highlighted in Fig. 6.(e), it remains evident that some inaccurate values persist without transitioning to accurate values, particularly in edge regions where inaccuracies are more prevalent. As previously explained, this phenomenon can occur with HMMF-based disparity refinement processors due to the relatively small number of pixels utilized in the process. Therefore, during visual confirmation, it becomes apparent that the proposed disparity refinement processor demonstrates superior performance when compared to conventional disparity refinement processors, as highlighted in Fig.6.(f).

Following visual confirmation, numerical analysis is indeed crucial. In Fig. 7–8, we present the experimental results obtained using the proposed and conventional disparity refinement processors for the KITTI 2012 and 2015 stereo benchmark datasets. The KITTI stereo benchmark datasets consist of two versions, namely 2012 and 2015. Consequently, we conducted separate experiments on both the KITTI 2012 and 2015 dataset versions.

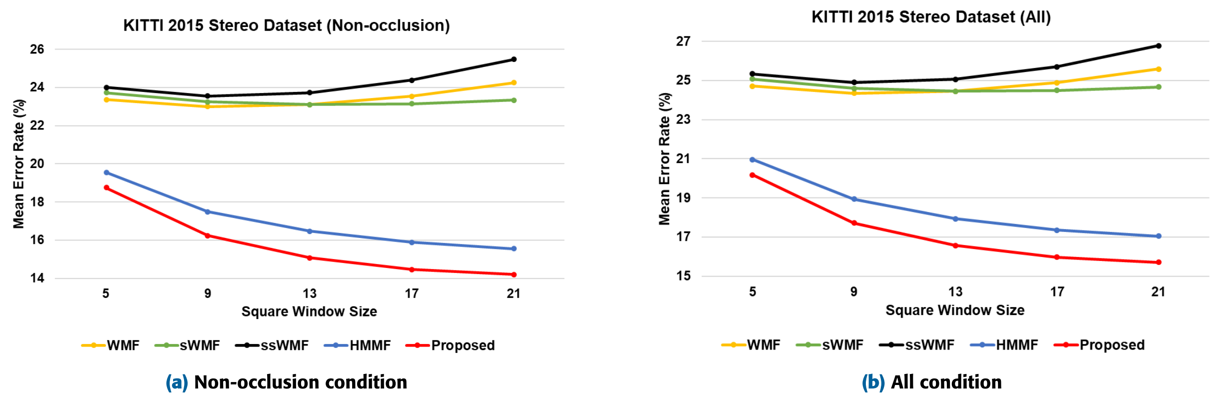


FIGURE 8. MER performance of the proposed and conventional disparity refinement processors when using the KITTI 2015 stereo benchmark dataset.

When utilizing the KITTI 2012 stereo benchmark dataset, as depicted in Fig. 7, the conventional WMF-based disparity refinement processor exhibited a high mean error rate (MER) index value. A high MER index value indicates lower matching accuracy performance. For both non-occlusion and all conditions, the proposed disparity refinement processor outperformed the conventional disparity refinement processors across all window sizes. Specifically, when using a 5 × 5 window size, the proposed processor demonstrated MER index values of 14.69% and 16.66% under non-occlusion and all conditions, respectively. In terms of performance improvement for numerical evaluation, the proposed processor enhanced refinement performance by 23.54% and 20.97% under non-occlusion and all conditions, respectively, compared to the conventional WMF-based processors. When compared to the HMMF-based processor, the proposed processor improved refinement performance by 4.20% and 3.64% under non-occlusion and all conditions, respectively. With a 21 × 21 window size, the proposed processor achieved MER index values of 12.12% and 14.15% under non-occlusion and all conditions, respectively. In comparison to WMF-based processors, the proposed processor improved refinement performance by 42.07% and 37.82%, respectively. Compared to the HMMF-based processor, the proposed processor improved refinement performance by 4.17% and 3.52%, respectively. In terms of the best improvement performance, our proposed processor enhanced refinement performance by 6.14% and 5.21% under non-occlusion and all conditions, respectively, when using a 13 × 13 window size.

When using the KITTI 2015 stereo benchmark dataset, as shown in Fig. 8, the conventional WMF-based disparity refinement processors also exhibited high MER index values, similar to the experimental results with the KITTI 2012 stereo benchmark dataset. In the experimental results using the KITTI 2015 stereo benchmark, our proposed disparity refinement processor also demonstrated superior refinement performance compared to the conventional WMF-based and HMMF-based disparity refinement processors across all window sizes. In terms of the performance improvement for numerical evaluation, when using a 13 × 13 window size, our proposed processor improved refinement performance by 36.45% and 33.89% under non-occlusion and all conditions, respectively, compared to the conventional WMF-based processors. When compared to the HMMF-based processor using the same window size, the proposed processor improved disparity refinement performance by 8.43% and 7.60% under non-occlusion and all conditions, respectively. In the best case, when using a 17 × 17 window size, the proposed processor enhanced disparity refinement performance by 40.65% and 37.86% under non-occlusion and all conditions, respectively, compared to the WMF-based processors. When compared with the HMMF-based processor using the same window size, the proposed processor improved refinement performance by 8.91% and 8.01% under non-occlusion and all conditions, respectively.

2) DRIVINGSTEREO

In the analysis of experimental results using the KITTI 2012 and 2015 stereo benchmark datasets, it was evident that the proposed disparity refinement processor consistently outperformed conventional disparity refinement processors across all window sizes. However, in terms of the KITTI 2012 and 2015 stereo benchmark datasets, they have relatively limited test stereo images, comprising 194 and 200 images, respectively. Therefore, for a more rigorous and objective performance assessment, it was imperative to employ a more extensive dataset than the KITTI stereo benchmark datasets. Consequently, we opted to incorporate the DrivingStereo dataset into our experiments. Released in 2019, this dataset boasts a substantial collection of 7751 test images [34].

Fig. 9 presents the experimental outcomes when employing the proposed and conventional disparity refinement processors with the Cityscapes dataset. Concerning the WMF-based disparity refinement processors, they displayed a notable trend of increasing MER index values as the window size grew larger. This increase in MER index values can be attributed to the fact that the initial disparity map generated using SGM often contains a significant number of

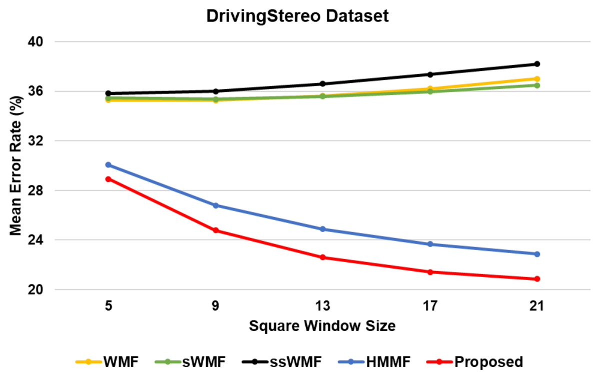


FIGURE 9. MER performance of the proposed and conventional disparity refinement processors when using the DrivingStereo dataset.

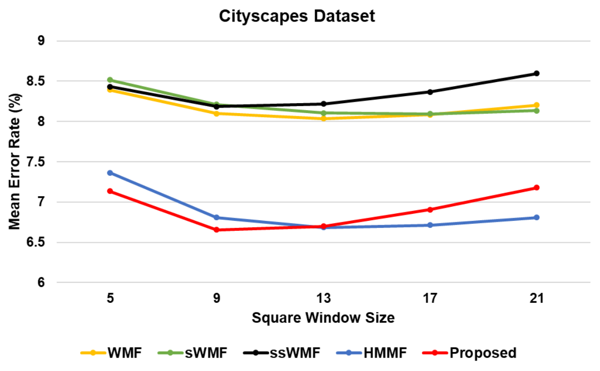


FIGURE 10. MER performance of the proposed and conventional disparity refinement processors when using the Cityscapes dataset.

hole regions. Consequently, WMF-based processors, which attempt to fill these hole regions based on weight parameters, can erroneously produce result values by mistaking incorrect values (computed as 0) within the hole region as correct values.

In contrast, both the proposed and HMMF-based disparity refinement processors exhibited a decreasing trend in the MER index value as the window size increased. When comparing the proposed processor to the HMMF-based processor, the proposed processor consistently delivered superior refinement performance across all window sizes. Specifically, with a 5 × 5 window size, the proposed processor achieved an MER index value of 28.93%. In comparison to conventional processors, the proposed processor exhibited improvements in disparity refinement performance of 18.03%, 18.43%, 19.26%, and 3.76% when compared to WMF, sWMF, ssWMF, and HMMF, respectively. When utilizing a 21 × 21 window size, the proposed processor enhanced disparity refinement performance by 43.68%, 42.86%, 45.43%, and 8.86% compared to WMF, sWMF, ssWMF, and HMMF, respectively. Based on the results from experiments conducted on the DrivingStereo dataset, it is evident that the proposed processor consistently outperforms conventional processors in terms of disparity refinement performance.

3) CITYSCAPES

To assess the disparity refinement performance in different driving environments, we conducted experiments using the Cityscapes dataset [33]. The Cityscapes dataset comprises 1525 stereo test images captured in various road environments across German cities such as Berlin, Leverkusen, Mainz, and Munich. This dataset provides an opportunity to evaluate the performance of disparity refinement in diverse driving scenarios, distinct from those encountered in the Chinese-collected DrivingStereo dataset. The Cityscapes dataset offers a wide range of driving environments within German cities, further enhancing our ability to evaluate the disparity refinement performance across a variety of scenarios, which differ from those presented in the DrivingStereo dataset.

Fig. 10 showcases the experimental results when employing the Cityscapes dataset for both the proposed and conventional disparity refinement processors In the case of the WMF-based disparity refinement processor, it consistently displayed high MER index values across all window sizes, similar to the observations made in the experiments using the KITTI 2012 and 2015 stereo benchmark datasets. Conversely, the proposed and HMMF-based disparity refinement processors consistently exhibited lower MER index values when compared to the WMF-based processors. Specifically, when comparing the proposed processor to the HMMF-based processor, the proposed disparity refinement processor consistently demonstrated superior refinement performance in window sizes ranging from 5 × 5 to 13 × 13. However, with larger window sizes (17 × 17 and 21 × 21), the proposed disparity refinement processor exhibited higher MER index values than the HMMF-based processor. Considering the experimental results from both the KITTI stereo benchmark datasets and the Cityscapes dataset, it is advisable to prioritize the use of smaller window sizes below 13 × 13.

B. ROBUSTNESS EVALUATION

To confirm the robustness of the proposed processor's refinement performance across various stereo matching algorithms, additional experiments were conducted. Initially, experiments were conducted using different disparity range and uniqueness threshold values of the disparitySGM built-in function to assess the robustness of our proposed processor. For this, a 13 × 13 window size was employed for both conventional and proposed processors. Table 1 illustrates the disparity refinement performance of both processors when varying the disparity range values from 64 to 128 on the KITTI 2015 stereo benchmark dataset. Given that the disparity range typically falls within the range of 64 to 128, the experiment utilized values of 64, 80, 96, and 128 [35], [36], [37], [38], [39], [40], [41]. Notably, across all disparity range values, our proposed processor demonstrated superior refinement performance under both all and non-occlusion conditions. These results highlight that the proposed processor consistently outperforms conventional processors, with the

TABLE 1. Refinement performance of the proposed and conventional processors at various disparity range values on the KITTI 2015 stereo benchmark dataset.

Disparity Range	Architecture	Mean Error Rate (%)	
		all (occlusion)	non-occlusion
128	WMF	23.0948	24.4441
	sWMF	23.1033	24.4526
	ssWMF	23.7251	25.0635
	HMMF	16.4647	17.9318
	Proposed	15.0774	16.5694
96	WMF	21.2877	22.6683
	sWMF	21.2974	22.6778
	ssWMF	21.9076	23.2774
	HMMF	14.5457	16.0465
	Proposed	13.1239	14.6501
80	WMF	20.7983	22.1875
	sWMF	20.8078	22.1970
	ssWMF	21.4221	22.8005
	HMMF	14.0129	15.5232
	Proposed	12.5782	14.1142
64	WMF	21.8025	23.1596
	sWMF	21.8028	23.1599
	ssWMF	22.4292	23.7757
	HMMF	15.0580	16.5338
	Proposed	13.6074	15.1088

TABLE 2. Refinement performance of the proposed and conventional processors at various uniqueness threshold values on the KITTI 2015 stereo benchmark dataset.

Threshold	Architecture	Mean Error Rate (%)	
		all (occlusion)	non-occlusion
5	WMF	20.7983	22.1875
	sWMF	20.8078	22.1970
	ssWMF	21.4221	22.8005
	HMMF	14.0129	15.5232
	Proposed	12.5782	14.1142
4	WMF	19.6179	21.0283
	sWMF	19.6265	21.0368
	ssWMF	20.1596	21.5607
	HMMF	13.6057	15.1234
	Proposed	12.3033	13.8444
3	WMF	18.6109	20.0393
	sWMF	18.6195	20.0479
	ssWMF	19.0929	20.5131
	HMMF	13.2736	14.7973
	Proposed	12.0758	13.6210
2	WMF	17.7759	19.2193
	sWMF	17.7812	19.2244
	ssWMF	18.2337	19.6692
	HMMF	12.9984	14.5270
	Proposed	11.8906	13.4393
1	WMF	17.0932	18.5488
	sWMF	17.0984	15.5539
	ssWMF	17.5486	18.9963
	HMMF	12.7697	14.3025
	Proposed	11.7358	13.2873

best performance observed at a disparity range value of 80.

Subsequently, additional experiments were conducted to compare the refinement performance at a disparity range value of 80, varying the uniqueness threshold value between one and five on the KITTI 2015 stereo benchmark dataset. The results, presented in Table 2, reaffirm the robustness of our proposed processor, which consistently exhibits superior refinement performance across all uniqueness threshold values, further confirming its robust characteristics when used in conjunction with the SGM algorithm for computing the initial disparity map.

To further verify the robustness characteristic of our proposed processor, we conducted additional experiments using MobileStereoNet [42]. Table 3 presents the refinement performance under all and non-occlusion conditions, respectively, when using both conventional and our proposed processor for the initial disparity map, employing MobileStereoNet on the KITTI 2015 stereo benchmark dataset. As indicated in the experimental results for the non-occlusion condition, the sWMF processor exhibited the best MER performance among the conventional processors, with a value of 4.7309%. Conversely, our proposed processor

TABLE 3. Refinement performance comparison using proposed and conventional disparity refinement processors on the KITTI 2015 stereo benchmark dataset with MobileStereoNet.

Work	Mean Error Rate (%)		Improvement (%)	
	all (occlusion)	non-occlusion	all (occlusion)	non-occlusion
WMF	4.8372	4.8646	3.87	4.24
sWMF	4.6991	4.7309	1.04	1.11
ssWMF	4.7405	4.7729	1.91	1.98
HMMF	4.7592	4.7871	2.29	2.27
Our	4.6501	4.6785	-	-

demonstrated a MER performance of 4.6785%. In terms of the all condition, the sWMF processor achieved a MER performance of 4.6991%, while our proposed processor exhibited a MER performance of 4.6501%. Numerically, our proposed processor improved the refinement performance by at least 1.11% and 1.04% under non-occlusion and all conditions, respectively, compared to the conventional processors. Based on the experimental results using MobileStereoNet, it can be concluded that our proposed processor possesses robustness characteristics.

C. HARDWARE PERFORMANCE COMPARISON

1) RESOURCE UTILIZATION

To ensure a fair comparison of hardware resource utilization, we employed the same FPGA board. Therefore, the proposed disparity refinement processor was synthesized on the Xilinx XC7K325T FPGA board, as previous studies had used this board and reported results.

For the sake of a fair comparison, both the proposed and conventional processors operated at a frequency of 148.5 MHz, utilized a disparity range of 128, and processed images at Full HD (FHD) resolution, equivalent to 1080p. Table 4 provides an overview of the synthesis results for the proposed and conventional disparity refinement processors. Notably, the WMF architecture was excluded from the comparison due to its significant resource requirements, as previously demonstrated in other works.

In Table 4, we observe the synthesis results of the proposed and conventional disparity refinement processors when implemented on an FPGA. Among the conventional disparity refinement processors, it is evident that the sWMF and ssWMF architectures demanded a substantial amount of hardware resource utilization. WMF-based architectures inherently involve a sorting process based on bilateral weight computation, which necessitates a considerable amount of BRAM utilization since dual-port BRAM must be used for sorting. In contrast, the HMMF architecture, which is our previous work, required fewer hardware resources than the WMF-based architectures. The reason behind the lower hardware resource utilization of HMMF is its avoidance of the bilateral weight-based sorting process. Consequently, HMMF exhibits cost-efficiency compared to the WMF-based architectures.

Among conventional disparity refinement processors, it was confirmed that HMMF, which omits the bilateral weight-based sorting process, incurred the least amount of hardware resource consumption. To assess whether the proposed disparity refinement processor also possesses cost-efficiency, we focused on comparing it with the HMMF synthesis results. Regarding BRAM, our proposed processor requires 19, 20, and 21 when the target square window sizes were 37, 39, and 41, respectively. The consistent BRAM utilization is due to the identical number of line buffers used for the disparity refinement process, which is based on the window size.

Regarding Slice Look-Up Tables (LUTs), our proposed processor requires 1327, 1433, and 1635 when the target square window sizes were 37, 39, and 41, respectively. For the reduction percentage of resource utilization, when using square window sizes of 37, 39, and 41, our proposed processor reduces resource utilization by 45.57%, 48.02%, and 49.57% compared to HMMF, respectively. In terms of Slice Registers, our proposed processor requires 2307, 2493, and 2599 when the target square window sizes were 37, 39, and 41, respectively. For the reduction percentage of resource utilization, when using square window sizes of 37, 39, and 41, our proposed processor reduces resource utilization by 32.58%, 35.09%, and 41.41% compared to HMMF, respectively. In the proposed disparity refinement processor, the window is generated only for the horizontal direction using three selected vertical maximum values. As a result, the proposed processor requires $(3 \times N)$ registers for the window generation step. In contrast, the HMMF architecture implements the window generation step in the Window Generator module based on the line buffers, leading to relatively higher slice register and LUT utilization. In conclusion, based on the synthesis results, we have confirmed that our proposed disparity refinement processor exhibits cost-efficiency when using large window sizes.

TABLE 4. Hardware resource utilization of the proposed and conventional disparity refinement processors when synthesized using window sizes ranging from 37 × 37 to 41 × 41 on Xilinx XC7K325T FPGA.

Window Size	Architecture	Resource Type		
		Slice Look-up Table (LUT)	Slice Register	Block Random Access Memory (BRAM)
41×41	sWMF [27]	Not provided	Not provided	Not provided
	ssWMF [28]	9737	5349	63
	HMMF [29]	3242	4436	21
	Proposed	1635	2599	21
39×39	sWMF [27]	12200	15813	55
	ssWMF [28]	Not provided	Not provided	Not provided
	HMMF [29]	2757	3840	20
	Proposed	1433	2493	20
37×37	sWMF [27]	Not provided	Not provided	Not provided
	ssWMF [28]	8211	4832	57
	HMMF [29]	2438	3422	19
	Proposed	1327	2307	19

In typical FPGA-based stereo vision systems using SGM, small-sized windows are commonly favored over large-sized windows, as shown in Table 5. To investigate this further, we conducted additional experiments to compare the required hardware resource utilization when implemented on an FPGA, focusing on small window sizes ranging from 5×5 to 21×21.

Table 5 presents the synthesis results for the proposed, HMMF-based, and ssWMF-based disparity refinement processors. In the case of the sWMF-based disparity refinement processor, previous studies did not provide experiment results on hardware resource consumption for square window sizes between 5 and 21. Consequently, our additional experiments focused on comparing the proposed processor with ssWMF-based and HMMF-based processors, excluding the sWMF-based one.

Regarding BRAM usage, the proposed disparity refinement processors requires the same number of $(N + 1)/2$ BRAMs as the HMMF-based disparity refinement processor, where N represents the required number of lines for generating predefined windows. Consequently, the proposed processor utilizes the same number of BRAMs as the HMMF-based processor. When compared to the WMF-based disparity refinement processor, our proposed processor demonstrated relatively lower hardware resource utilization.

Regarding slice LUTs and slice registers, both the proposed and conventional disparity refinement processors show a linear increase with larger window sizes. However, in the case of the ssWMF-based disparity refinement processor, it inherently demands a substantial amount of hardware resources, even with a small window size, due to the bilateral weight-based sorting process. In contrast, the HMMF-based disparity refinement processor requires fewer hardware resources compared to the ssWMF-based one. When comparing the proposed processor with the HMMF-based processor, our proposed disparity refinement processor consistently demonstrates lower hardware resource utilization, consistent with the experimental results presented in Table 4. This is attributed to the fact that, as explained earlier, our proposed processor, unlike the HMMF-based processor, generates only the minimum required registers for the window after the maximum value selection operation, resulting in relatively excellent optimization.

2) POWER CONSUMPTION

When using Xilinx software to implement on an FPGA, it automatically provides the expected power consumption. Fig. 11 displays the power consumption report of the proposed processor when operating at frequencies of 74.25 MHz and 148.5 MHz for square window sizes ranging from 37 to 41. At an operating frequency of 74.25 MHz, the proposed processor exhibited power consumptions of 189 mW, 192 mW, and 196 mW for square window sizes of 37, 39, and 41, respectively. When operating at 148.5 MHz, the power consumptions were 270 mW, 276 mW, and 284 mW for the same window sizes. In comparison, conventional SGM-based stereo vision systems typically demand total power ranging from at least 2.31 W to 9.8 W [9], [36], [39], [43], [44], [45]. Therefore, when considering

TABLE 5. Hardware resource utilization of the proposed and conventional disparity refinement processors when synthesized using window sizes ranging from 5 × 5 to 21 × 21 on Xilinx XC7K325T FPGA.

		Resource Type		
Window Size	Architecture	Slice LUT	Slice Register	BRAM
21 × 21	ssWMF [28]	4046	2565	33
	HMMF [29]	1165	1835	11
	Proposed	712	1437	11
17 × 17	ssWMF [28]	3035	2105	27
	HMMF [29]	839	1344	9
	Proposed	581	1086	9
13 × 13	ssWMF [28]	2040	1516	21
	HMMF [29]	773	1265	7
	Proposed	453	925	7
9 × 9	ssWMF [28]	1400	1096	15
	HMMF [29]	535	889	5
	Proposed	326	653	5
5 × 5	ssWMF [28]	692	656	9
	HMMF [29]	352	598	3
	Proposed	198	457	3

the total power of conventional SGM-based stereo vision systems, the power consumption of our proposed processor is relatively low. To further compare power consumption with conventional disparity refinement processors, we attempted to find experimental reports from previous works. "However, we were unable to locate power consumption reports in the existing research literature." Therefore, we abstain from directly comparing the power consumption performance of our proposed processor to conventional processors.

3) PROCESSING PERFORMANCE

To facilitate a comprehensive comparison of processing speed and computational performance, we assess two critical factors: 1) frame per second (FPS) and 2) million disparity estimation per second (MDE/s). The computation of MDE/s is illustrated by (19).

$$MDE/s = (W \times H) \times DR \times FPS. \tag{19}$$

where W and H denote the width and height of the disparity map, thus $(W \times H)$ represents the resolution of the input image. DR stands for the disparity range, and FPS indicates the frame rate. In essence, the MDE/s factor provides insight into the number of disparity values that can be processed in one second, considering substantial quantities.

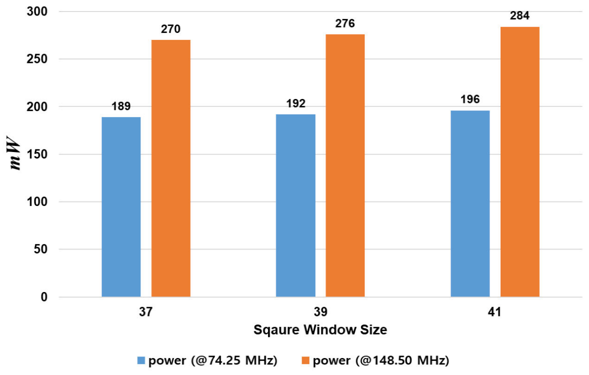


FIGURE 11. Power consumption of our proposed processor at operation frequencies of 74.25 MHz and 148.5 MHz.

Table 6 outlines the processing performance of both the conventional and proposed disparity refinement processors. At FHD resolution, a frame rate of 60 FPS is achievable with an operational frequency of 148.5 MHz. In practical terms, real-time operation is assured when the maximum operating frequency exceeds 148.5 MHz. Consequently, we conducted an analysis of the maximum operating frequency for our proposed processor and confirmed its operation at frequencies of up to 273.97 MHz. Thus, when considering the use of FHD resolution, our proposed processor can operate in real-time, meeting the requirement of a working frequency of at least 148.5 MHz.

By determining the maximum operating frequency, the corresponding frames per second (FPS) can be derived. Our proposed processor demonstrated a high processing speed, achieving 110 FPS, surpassing the performance of conventional processors. In a direct arithmetic comparison, the FPS of the proposed processor exhibited superior results, ranging from a minimum of 1.36 times to a maximum of 1.83 times compared to conventional processors. Therefore, when using the same disparity range value, such as 128, the arithmetic comparison results of the MDE/s align with the arithmetic comparison results of the FPS, as dictated by Equation (19). This robust performance underscores the efficiency and computational prowess of the proposed processor in handling large quantities of disparity values in real-time scenarios.

To evaluate the processing performance of the proposed processor for the SGM algorithm, additional experiments were conducted, as the processor was specifically designed for SGM-based stereo vision systems. Table 7 presents the processing performance of our implemented SGM-based stereo vision system compared to conventional stereo vision systems. In the FPGA category, numerous prior works have aimed to achieve at least one of three goals for real-time operation: 1) improving processing speed, 2) enhancing matching accuracy, and 3) reducing hardware resource utilization [39], [43], [46], [47], [48], [49], [50], [51].

The experimental results in Table 7 confirm the feasibility of real-time processing on the FPGA platform. For this reason, we implemented the SGM algorithm on the FPGA

TABLE 6. Comparison of processing performance for conventional and proposed disparity refinement processors.

Architectures	Resolution	DR [1]	Maximum Operating Freaking (MHz)	FPS [2]	MDE/s [3]
sWMF [27]	1920 × 1080	128	148.50	60	15925
ssWMF [28]	1920 × 1080	128	167.95	67	17783
HMMF [29]	1920 × 1080	128	202.75	81	21499
Proposed	1920 × 1080	128	273.97	110	29196

[1] DR: Disparity Range
[2] FPS: Frame Per Second
[3] MDE/s: Million Disparity Estimation per Second

TABLE 7. Comparison of processing performance for various FPGA-based and GPU-based stereo vision systems.

Category	Work	Platform	Resolution	DR	FPS	Description
FPGA	Banz, et al. [39]	Virtex-5	640 × 480	128	103	SGM
	Qamar, et al. [46]	Virtex-7	640 × 480	128	30	SGM
	Mattoccia, et al. [47]	Spartan-6	640 × 480	32	30	Census
	Cambuim, et al. [43]	Cyclone-4	1024 × 768	128	127	SAD + SGM
	Zhang, et al. [48]	Altera	1024 × 768	64	60	Mini-Census
	Shan, et al. [49]	Stratix-4	1024 × 768	128	129	MCADSR
	Puglia, et al. [50]	Virtex-7	1024 × 768	64	30	DP
	Ttofis, et al. [51]	Kintex-7	1280 × 720	64	60	Guided Filter
	Our	Kintex-7	1280 × 720	128	72	SGM + LRC + Proposed
GPU	Chang, et al. [53]	NVIDIA Jetson TX2	1242 × 375	64	12	StereoVAE
	Chang, et al. [53]	NVIDIA Jetson AGX Xavier	1242 × 375	64	30	StereoVAE
	Aleotti, et al. [54]	NVIDIA Jetson TX2	1280 × 384	-	$\simeq 1$	DWARF
	Aleotti, et al. [54]	NVIDIA GTX 1080Ti	1280 × 384	-	$\simeq 10$	DWARF
	Chang, et al. [55]	NVIDIA Jetson TX2	1280 × 384	128	32	ZNCC
	Chang, et al. [56]	NVIDIA GTX 780Ti	1444 × 960	380	30	-
	Chang, et al. [56]	NVIDIA GTX 1080Ti	1444 × 960	380	72	-

platform using only the programmable logic region. To ensure real-time operational performance, we adopted the 4-path, including north, north-west, west, and west-south directions, for directional cost operation processes [40], [52]. The use of the 4-path directional cost allowed for a hardware-friendly implementation using only the programmable logic (PL) region on the FPGA. In the post-processing stage, the left-right consistency check (LRC) and the proposed processor were employed. Synthesizing our SGM-based stereo vision system, it was confirmed that a frame rate of 72 FPS is achievable when using HD resolution.

On the other hand, when considering GPU-based stereo vision systems, they have demonstrated high matching accuracy performance in previous works [53], [54], [55], [56]. However, as indicated by the FPS values in Table 7, it is evident that real-time performance cannot be guaranteed in the case of layer-based stereo matching algorithms, despite leveraging GPU capabilities. In other words, on low-cost embedded platforms without GPUs, it becomes challenging to employ layer-based stereo matching algorithms, which may exhibit high accuracy but struggle to achieve real-time processing. Therefore, for real-time applications such as

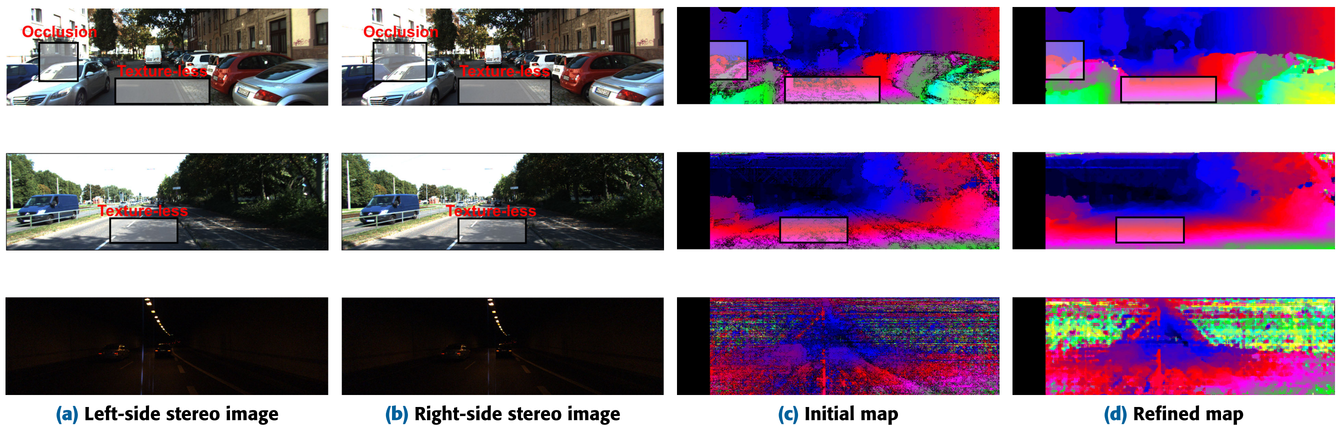


FIGURE 12. Experimental results for analyzing images in occluded and texture-less regions.

ADAS, the use of an SGM-based embedded stereo vision system becomes a viable and practical option.

VI. DISCUSSION

In this study, we presented various experimental results of the proposed processor when used with SGM and MobileStereoNet. When employing MobileStereoNet, the refinement performance of the proposed processor was found to be better than that of conventional processors, although the improvement was not substantial. This is attributed to the fact that deep learning-based stereo matching, such as MobileStereoNet, inherently exhibits high initial map accuracy performance.

In other words, accurate disparity values can be computed in both texture-less and occluded regions when using deep learning-based stereo matching. On the other hand, when using SGM, holes that cannot be computed occur in texture-less and occluded regions in the initial disparity map, as illustrated in Fig. 12. Traditional approach-based stereo matching, like SGM, often generates clustered hole regions because it computes disparity values by performing feature matching using a general window shape or a window shape based on the disparity range.

Analyzing specific regions, as depicted in Fig. 12.(c), occluded regions typically arise when distant objects are obscured by foreground elements from the reference point where the frame was captured. Consequently, occluded regions lead to disparity discontinuity, and the disparity value in the occluded region must be computed based on the occluded target object. In texture-less regions, often termed monotonous regions, finding the disparity value using traditional approach-based stereo matching is challenging, as illustrated in Fig. 12.(c), due to the difficulty in identifying the target matching point in these regions.

When the proposed processor is used in the refinement process, it effectively replaces clustered hole regions in occluded and texture-less areas with correct values, as demonstrated in Fig. 12.(d). This efficacy is attributed to the proposed processor's ability to reflect disparity characteristics in both the vertical and horizontal directions within the driving road environment.

In the case of the color domain, analysis can be conducted through the results of the left and right-side stereo images shown in the third row of Fig. 12. Traditional approach-based stereo matching relies on feature points, called matching points, and assumes that objects have a discernible shape. However, in environments such as tunnels, where objects are not clearly visible, stereo matching may not proceed smoothly. Stereo matching in low-light environments requires appropriate pre-processing techniques to convert the image from low-dynamic range to high-dynamic range for effective performance.

VII. CONCLUSION

In this study, we introduced a novel cell-based disparity refinement processor designed for the disparity refinement process in stereo vision systems. To evaluate the performance of our proposed processor, we conducted comprehensive experiments using three distinct public stereo datasets. When we applied our disparity refinement processor to the KITTI 2012/2015 stereo benchmark datasets and the DrivingStereo dataset across a range of window sizes, the results consistently showed that our processor outperformed traditional methods. Notably, when we utilized the Cityscapes dataset with window sizes ranging from 5×5 to 13×13, our processor demonstrated superior performance compared to conventional approaches. Collectively, our experimental findings from these three public stereo datasets indicate that our proposed disparity refinement processor consistently delivers outstanding disparity refinement performance, especially when employed with window sizes under 13×13.

Regarding hardware resource utilization on an FPGA, our study revealed that our proposed disparity refinement processor requires fewer hardware resources than conventional methods for all window sizes. When compared to WMF-based disparity refinement processors, our processor's simpler architecture, relying solely on max and median filters, results in reduced hardware resource consumption.

Furthermore, in contrast to HMMF-based disparity refinement processors, our approach eliminates the need for a window generation step when computing the output value of the refined disparity map. This optimized design positions our proposed disparity refinement processor favorably. In conclusion, based on our experimental results, the proposed disparity refinement processor offers an appealing combination of low-cost and high-performance characteristics compared to conventional methods, particularly when employed with window sizes below 13 × 13.

For future work, we intend to conduct a detailed analysis to understand why our proposed disparity refinement processor exhibited relatively lower refinement performance compared to HMMF-based processors when using larger window sizes (17 × 17 and 21 × 21) on the Cityscapes dataset. Based on this analysis, we plan to explore optimized methods and hardware architectures to further enhance disparity refinement performance. Additionally, we aim to conduct experiments with infrared stereo cameras, such as QuantumRed provided by Hanwha Systems, Co., Ltd., as well as YUV or RGB-based stereo cameras frequently used in autonomous driving scenarios.

REFERENCES

[1] J. Zhang, T. Yang, Q. Li, B. Zhou, Y. Yang, G. Luo, and J. Shi, "An FPGA-based neural network overlay for ADAS supporting multi-model and multi-mode," in *Proc. IEEE Int. Symp. Circuits Syst. (ISCAS)*, May 2021, pp. 1–5.

[2] I. Raouf, A. Khan, S. Khalid, M. Sohail, M. M. Azad, and H. S. Kim, "Sensor-based prognostic health management of advanced driver assistance system for autonomous vehicles: A recent survey," *Mathematics*, vol. 10, no. 18, p. 3233, Sep. 2022.

[3] L. Wang, P. Sun, M. Xie, S. Ma, B. Li, Y. Shi, and Q. Su, "Advanced driver-assistance system (ADAS) for intelligent transportation based on the recognition of traffic cones," *Adv. Civil Eng.*, vol. 2020, pp. 1–8, Jun. 2020.

[4] J. Li, L. Yao, X. Xu, B. Cheng, and J. Ren, "Deep reinforcement learning for pedestrian collision avoidance and human–machine cooperative driving," *Inf. Sci.*, vol. 532, pp. 110–124, Sep. 2020.

[5] M. Seyedi, M. Koloushani, S. Jung, and A. Vanli, "Safety assessment and a parametric study of forward collision-avoidance assist based on real-world crash simulations," *J. Adv. Transp.*, vol. 2021, Dec. 2021, Art. no. 4430730.

[6] Y. Yuan, Y. Lu, and Q. Wang, "Adaptive forward vehicle collision warning based on driving behavior," *Neurocomputing*, vol. 408, pp. 64–71, Sep. 2020.

[7] G. Gao, L. Wei, and Q. Hu, "Research on image lightweight binocular ranging," in *Proc. 3rd Int. Conf. Comput., Control Robot. (ICCCR)*, Mar. 2023, pp. 82–86.

[8] J. Watson, M. Firman, G. Brostow, and D. Turmukhambetov, "Self-supervised monocular depth hints," in *Proc. IEEE/CVF Int. Conf. Comput. Vis. (ICCV)*, Oct. 2019, pp. 2162–2171.

[9] L. F. S. Cambuim, L. A. Oliveira, E. N. S. Barros, and A. P. A. Ferreira, "An FPGA-based real-time occlusion robust stereo vision system using semi-global matching," *J. Real-Time Image Process.*, vol. 17, no. 5, pp. 1447–1468, Oct. 2020.

[10] J. Toledo, M. Lauer, and C. Stiller, "Real-time stereo semi-global matching for video processing using previous incremental information," *J. Real-Time Image Process.*, vol. 19, no. 1, pp. 205–216, Feb. 2022.

[11] Y.-Q. Guo, M. Gu, and Z.-D. Xu, "Research on the improvement of semi-global matching algorithm for binocular vision based on lunar surface environment," *Sensors*, vol. 23, no. 15, p. 6901, Aug. 2023.

[12] J. Wang, K. Gong, T. Balz, N. Haala, U. Soergel, L. Zhang, and M. Liao, "Radargrammetric DSM generation by semi-global matching and evaluation of penalty functions," *Remote Sens.*, vol. 14, no. 8, p. 1778, Apr. 2022.

[13] M. Jang, H. Yoon, S. Lee, J. Kang, and S. Lee, "A comparison and evaluation of stereo matching on active stereo images," *Sensors*, vol. 22, no. 9, p. 3332, Apr. 2022.

[14] Z. Lu, J. Wang, Z. Li, S. Chen, and F. Wu, "A resource-efficient pipelined architecture for real-time semi-global stereo matching," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 32, no. 2, pp. 660–673, Feb. 2022.

[15] Z. Wan, B. Yu, T. Y. Li, J. Tang, Y. Zhu, Y. Wang, A. Raychowdhury, and S. Liu, "A survey of FPGA-based robotic computing," *IEEE Circuits Syst. Mag.*, vol. 21, no. 2, pp. 48–74, 2nd Quart., 2021.

[16] X. Xiang, M. Zhang, G. Li, Y. He, and Z. Pan, "Real-time stereo matching based on fast belief propagation," *Mach. Vis. Appl.*, vol. 23, no. 6, pp. 1219–1227, Nov. 2012.

[17] S.-F. Hsiao, W.-L. Wang, and P.-S. Wu, "VLSI implementations of stereo matching using dynamic programming," in *Proc. Tech. Papers Int. Symp. VLSI Design, Autom. Test*, Apr. 2014, pp. 1–4.

[18] B. Lu, L. Sun, L. Yu, and X. Dong, "An improved graph cut algorithm in stereo matching," *Displays*, vol. 69, Sep. 2021, Art. no. 102052.

[19] R. Szeliski, R. Zabih, D. Scharstein, O. Veksler, V. Kolmogorov, A. Agarwala, M. Tappen, and C. Rother, "A comparative study of energy minimization methods for Markov random fields with smoothness-based priors," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 30, no. 6, pp. 1068–1080, Jun. 2008.

[20] D. Scharstein and R. Szeliski, "A taxonomy and evaluation of dense two-frame stereo correspondence algorithms," *Int. J. Comput. Vis.*, vol. 47, nos. 1–3, pp. 7–42, Apr. 2002.

[21] H. Hirschmuller, "Stereo processing by semiglobal matching and mutual information," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 30, no. 2, pp. 328–341, Feb. 2008.

[22] Y. Xie, S. Zeng, and L. Chen, "A novel disparity refinement method based on semi-global matching algorithm," in *Proc. IEEE Int. Conf. Data Mining Workshop*, Dec. 2014, pp. 1135–1142.

[23] C. Stentoumis, E. Karkalou, and G. Karras, "A review and evaluation of penalty functions for semi-global matching," in *Proc. IEEE Int. Conf. Intell. Comput. Commun. Process. (ICCP)*, Sep. 2015, pp. 167–172.

[24] Z. Ma, K. He, Y. Wei, J. Sun, and E. Wu, "Constant time weighted median filtering for stereo matching and beyond," in *Proc. IEEE Int. Conf. Comput. Vis.*, Dec. 2013, pp. 49–56.

[25] Q. Zhang, L. Xu, and J. Jia, "100+ times faster weighted median filter (WMF)," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, Jun. 2014, pp. 2830–2837.

[26] W. Wu, L. Li, and W. Jin, "Disparity refinement based on segment-tree and fast weighted median filter," in *Proc. IEEE Int. Conf. Image Process. (ICIP)*, Sep. 2016, pp. 3449–3453.

[27] S. Chen, X. Zhang, H. Sun, and N. Zheng, "SWMF: Separable weighted median filter for efficient large-disparity stereo matching," in *Proc. IEEE Int. Symp. Circuits Syst. (ISCAS)*, May 2017, pp. 1–4.

[28] J. Hyun, Y. Kim, J. Kim, and B. Moon, "Hardware-friendly architecture for a pseudo 2D weighted median filter based on sparse-window approach," *Multimedia Tools Appl.*, vol. 80, nos. 26–27, pp. 34221–34236, Nov. 2021.

[29] C.-H. Choi and H. W. Oh, "Dispairty refinement processor architecture utilizing horizontal and vertical characteristics for stereo vision systems," in *Proc. 26th Euromicro Conf. Digit. Syst. Des. (DSD)*, Sep. 2023, pp. 220–226.

[30] R. A. Schowengerdt, *Remote Sensing: Models and Methods for Image Processing*. Amsterdam, The Netherlands: Elsevier, 2006.

[31] A. Geiger, P. Lenz, and R. Urtasun, "Are we ready for autonomous driving? The KITTI vision benchmark suite," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, Jun. 2012, pp. 3354–3361.

[32] M. Menze and A. Geiger, "Object scene flow for autonomous vehicles," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2015, pp. 3061–3070.

[33] M. Cordts, M. Omran, S. Ramos, T. Rehfeld, M. Enzweiler, R. Benenson, U. Franke, S. Roth, and B. Schiele, "The cityscapes dataset for semantic urban scene understanding," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2016, pp. 3213–3223.

[34] G. Yang, X. Song, C. Huang, Z. Deng, J. Shi, and B. Zhou, "DrivingStereo: A large-scale dataset for stereo matching in autonomous driving scenarios," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2019, pp. 899–908.

[35] Z. Chen, P. Dong, Z. Li, R. Yao, Y. Ma, X. Fang, H. Deng, W. Zhang, L. Chen, and F. An, "Real-time FPGA-based binocular stereo vision system with semi-global matching algorithm," in *Proc. IEEE 34th Int. System-on-Chip Conf. (SOCC)*, Sep. 2021, pp. 158–163.

[36] 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.

[37] M. Roszkowski and G. Pastuszak, "FPGA design of the computation unit for the semi-global stereo matching algorithm," in *Proc. 17th Int. Symp. Design Diag. Electron. Circuits Syst.*, Apr. 2014, pp. 230–233.

[38] C. Ttofis and T. Theocharides, "High-quality real-time hardware stereo matching based on guided image filtering," in *Proc. Design, Autom. Test Eur. Conf. Exhib. (DATE)*, Mar. 2014, pp. 1–6.

[39] C. Banz, S. Hesselbarth, H. Flatt, H. Blume, and P. Pirsch, "Real-time stereo vision system using semi-global matching disparity estimation: Architecture and FPGA-implementation," in *Proc. Int. Conf. Embedded Comput. Systems: Architectures, Modeling Simulation*, Jul. 2010, pp. 93–101.

[40] S. K. Gehrig, F. Eberli, and T. Meyer, "A real-time low-power stereo vision engine using semi-global matching," in *Proc. Int. Conf. Comput. Vis. Syst. (ICVS)*. Berlin, Germany: Springer, 2009, pp. 134–143.

[41] Y. Li, Z. Li, C. Yang, W. Zhong, and S. Chen, "High throughput hardware architecture for accurate semi-global matching," *Integration*, vol. 65, pp. 417–427, Mar. 2019.

[42] F. Shamsafar, S. Woerz, R. Rahim, and A. Zell, "MobileStereoNet: Towards lightweight deep networks for stereo matching," in *Proc. IEEE/CVF Winter Conf. Appl. Comput. Vis. (WACV)*, Jan. 2022, pp. 677–686.

[43] 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 (SBCCI)*, Aug. 2017, pp. 53–58.

[44] O. Rahnama, T. Cavalleri, S. Golodetz, S. Walker, and P. Torr, "R3SGM: Real-time raster-respecting semi-global matching for power-constrained systems," in *Proc. Int. Conf. Field-Program. Technol. (FPT)*, Dec. 2018, pp. 102–109.

[45] J. Zhao, T. Liang, L. Feng, W. Ding, S. Sinha, W. Zhang, and S. Shen, "FP-Stereo: Hardware-efficient stereo vision for embedded applications," in *Proc. 30th Int. Conf. Field-Program. Log. Appl. (FPL)*, Aug. 2020, pp. 269–276.

[46] A. Qamar, F. B. Muslim, and L. Lavagno, "Analysis and implementation of the semi-global matching 3D vision algorithm using code transformations and high-level synthesis," in *Proc. IEEE 81st Veh. Technol. Conf. (VTC Spring)*, May 2015, pp. 1–5.

[47] S. Mattoccia and M. Poggi, "A passive RGBD sensor for accurate and real-time depth sensing self-contained into an FPGA," in *Proc. 9th Int. Conf. Distrib. Smart Cameras*, Sep. 2015, pp. 146–151.

[48] 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*, Feb. 2011, pp. 55–64.

[49] 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. 4s, pp. 1–24, Jul. 2014.

[50] L. Puglia, M. Vigliar, and G. Raiconi, "Real-time low-power FPGA architecture for stereo vision," *IEEE Trans. Circuits Syst. II, Exp. Briefs*, vol. 64, no. 11, pp. 1307–1311, Nov. 2017.

[51] 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.

[52] D. Hernandez-Juarez, A. Chacón, A. Espinosa, D. Vázquez, J. C. Moure, and A. M. López, "Embedded real-time stereo estimation via semi-global matching on the GPU," *Proc. Comput. Sci.*, vol. 80, pp. 143–153, Jan. 2016.

[53] Q. Chang, X. Li, X. Xu, X. Liu, Y. Li, and J. Miyazaki, "StereoVAE: A lightweight stereo-matching system using embedded GPUs," in *Proc. IEEE Int. Conf. Robot. Autom. (ICRA)*, May 2023, pp. 1982–1988.

[54] F. Aleotti, M. Poggi, F. Tosi, and S. Mattoccia, "Learning end-to-end scene flow by distilling single tasks knowledge," in *Proc. AAAI Conf. Artif. Intell.*, 2020, vol. 34, no. 7, pp. 10435–10442.

[55] Q. Chang, A. Zha, W. Wang, X. Liu, M. Onishi, L. Lei, M. J. Er, and T. Maruyama, "Efficient stereo matching on embedded GPUs with zero-means cross correlation," *J. Syst. Archit.*, vol. 123, Feb. 2022, Art. no. 102366.

[56] Q. Chang and T. Maruyama, "Real-time stereo vision system: A multi-block matching on GPU," *IEEE Access*, vol. 6, pp. 42030–42046, 2018.

CHEOL-HO CHOI received the B.S. degree from the Department of Electronic Engineering, Yeungnam University, Gyeongsan-si, Republic of Korea, in 2020, and the M.S. degree in electronic and electrical engineering from Kyungpook National University, Daegu, Republic of Korea, in 2022. He is currently a member of the Pangyo Research and Development Center, Core H/W Team, and has been with Hanwha Systems Company Ltd., since 2023. He is also a Digital Circuit Design Engineer in image processor. He has authored some journal articles and presented papers at both national and international conferences, with a primary focus on hardware-friendly algorithm and processor architecture for domain-specific image processor. His current research interests include computer vision, hardware architecture, and VLSI design.

HYUN WOO OH received the B.S. degree in electronic and IT media engineering and the M.S. degree in electronic engineering from the Seoul National University of Science and Technology, Seoul, Republic of Korea, in 2021 and 2023, respectively. He is currently a member of the Pangyo Research and Development Center, Core H/W Team, and has been with Hanwha Systems Company Ltd., since 2023. He is also a SoC Design Engineer in image processor. He has authored some journal articles and presented papers at both national and international conferences, with a primary focus on domain-specific processor architectures and VLSI design. His current research interests include computer architecture, system on chip, and edge-oriented reconfigurable computing.

JOONHWAN HAN received the B.S. degree in electrical and electronic engineering from Kwangwoon University, Seoul, Republic of Korea, in 2003, and the M.S. degree from the Department of NCW, Ajou University, Suwon, Republic of Korea, in 2018. He is currently a member of the Pangyo Research and Development Center, Core H/W Team, and has been with Hanwha Systems Company Ltd., since 2002. He is also a Lead Researcher of the Infrared Technology Research and Development Project. His research interests include system architecture and image processing.

JUNGHO SHIN received the B.S. degree from the Department of Electronic Engineering, Kyunghee University, Yongin, Republic of Korea, in 2010. He is currently pursuing the M.S. degree with the Department of Electrical and Computer Engineering, Sungkyunkwan University, Suwon, Republic of Korea. Since 2010, he has been a member of the Pangyo Research and Development Center, Core H/W Team, and has been with Hanwha Systems Company Ltd. He is also a Circuit Design Engineer in infrared thermal systems. His current research interests include system architecture and image processing.