Real-time multi-window stereo matching algorithm with fuzzy logic

Héctor-Daniel Vázquez-Delgado | Madaín Pérez-Patricio | Abiel Aguilar-González | Miguel-Octavio Arias-Estrada | Marco-Antonio Palacios-Ramos | Jorge Luis Camas-Anzueto | Antonio Pérez-Cruz | Sabino Velázquez-Trujillo

1Tecnológico Nacional de México/IT de Tuxtla Gutiérrez, Tuxtla Gutiérrez, Chiapas, México
2Instituto Nacional de Astrofísica Óptica y Electrónica, Puebla, México

Correspondence
Héctor-Daniel Vázquez-Delgado, Carretera panamericana km. 1080, Tuxtla Gutiérrez, 29050, Chiapas, México.
Email: mperez@ittg.edu.mx

Funding information
CONACyT

Abstract
Stereo matching obtains a depth map called a disparity map that indicates or shows the positions of the objects in a scene. To estimate a disparity map, the most popular trend consists of comparing two images (left-right) from two different points from the same scene. Unfortunately, small window sizes are suitable to preserve the edges, while large window sizes are required in homogeneous areas. To solve this problem, in this article, a novel real-time stereo matching algorithm embedded in an FPGA is proposed. The approach consists of estimating disparity maps with different window sizes by using the sum of absolute differences (SAD) as a local correlation metric. Once the disparity maps are obtained, the left-right consistency for each window size is computed. At the end of this stage, the centre pixel deviation is estimated through a $5 \times 5$ window and the Sobel gradient is extracted from the left image. Finally, both parameters are processed by a Fuzzy Inference System (FIS), which combines the calculated disparities and generates a final disparity map. An architecture embedded in FPGA is established and hardware acceleration strategies are discussed. Experimental results demonstrated that this algorithmic formulation provides promising results compared with the current state of the art.

1 | INTRODUCTION

Depth estimation is one of the most important tasks of the current computer vision systems and it is also the basis of several real-world applications such as positioning systems for mobile robotics [1], medical applications [2] and so on. Even though there are several approaches and resources to estimate depth in a scene, in order to do such action, stereo matching has become the most popular trend to estimate the depth from images obtained by a stereo configuration. For the stereo-matching approach, point correspondences between stereo pairs (two different viewpoints with epipolar constraint from the same scene) allow to obtain images of depth called disparity maps. In order to estimate a disparity map, it is necessary to measure the similarity of the points inside the stereo pair. The stereo matching process generally consists of the following four steps: (1) matching cost computation; (2) cost aggregation; (3) disparity computation/optimization; and (4) disparity refinement. In order to guarantee real-time processing, stereo-matching algorithms are often implemented using dedicated hardware such as FPGA [3] or CUDA technologies [4].

1.1 | Stereo-matching algorithms suitable for hardware implementation

The current state of the art has demonstrated that local algorithms reach promising results (in terms of processing speed and embedded capabilities) and are being implemented inside dedicated hardware [5,6]. As a result, in present study, local algorithms implemented inside FPGA or GPU architectures are the most popular solution for applications in which real-time processing and embedded capabilities are required [7–9];...
FPGA is the technology with a higher performance in terms of embedded capabilities [10,11].

For real-time processing and embedded capacities, several stereo-matching algorithms in the present study have been implemented using FPGA technology [3,12]. Depending on the configuration of the cameras, the range of disparity levels varies for algorithms implemented inside FPGA devices; this implies an increment of hardware resources consumption. This problem has motivated diverse authors to study the possibility of decreasing hardware resources usage while a large range of disparity levels for each pixel in the scene is estimated. However, this is still an open problem, therefore, several works in the current literature search for new approaches to implement ‘efficient’ stereo-matching algorithms inside FPGA devices [13].

1.2 | Motivation and scope

Over the last decade, several works have demonstrated that local stereo-matching algorithms implemented inside FPGA devices are highly useful under several real-world applications. Unfortunately, for high flexibility (regarding depth estimation) high range of disparity levels for each pixel in the scene is required but this implies high hardware resource consumption. Besides, estimation of disparity levels for each pixel in the scene demands the increase of hardware requirements. So, in this article, several investigations looked for new approaches to implement ‘efficient’ stereo-matching algorithms inside FPGA.

In this work, a novel real-time stereo matching algorithm embedded in a FPGA device was proposed. The approach consists of estimating disparity maps with different window sizes by using a simple local metric correlation. Then, a Fuzzy Inference System combined the previously computed disparities in order to generate a final disparity map. On the one hand, this disparity map used small correlation windows under object boundaries and on the other hand, it used large correlation windows for homogeneous areas. In order for the embedded capabilities to be guaranteed, an FPGA architecture was presented and hardware acceleration strategies were discussed.

Concerning the metric correlation, the use of the widely known Sum of Absolute Differences (SAD) was proposed as the local metric correlation. Advanced stereo matching techniques such as those based on semi-global matching or based on cross-based aggregation could outperform SAD in terms of accuracy, but here small system design is analysed (suitable for smart cameras), and to the authors’ knowledge, SAD provided a better trade-off between accuracy and hardware requirements. Furthermore, in order to guarantee low system requirements, the design of a parallel-pipeline architecture was presented. So, the main contributions were:

1. A novel stereo matching algorithm which used multiple correlation windows that guaranteed not only high performance in terms of accuracy but guaranteed embedded capabilities as well.

2. An FPGA architecture suitable for smart cameras.

2 | RELATED WORK

The ‘Middlebury Stereo Evaluation v.3’ is a popular dataset for stereo matching evaluation. In order to have a proper comparison baseline, only other consulted related works are presented on this section.

The system presented in [14] shows an architecture called Multi-Scale and Multi-Dimension network (MSMD). First, a new multi-scale equalisation costing subnet, in which two different receptive in parallel field sizes were implemented and developed. In addition, it was demonstrated that a multi-dimensional aggregation subnet containing 2D convolution and 3D convolution operations provided context-rich information and semantic information to estimate an accurate initial disparity.

In [15] a differentiable PatchMatch module that allows management of most disparities without requiring full cost volume evaluation is presented. The proposed representation (PatchMatch) was exploited to learn which range to prune for each pixel. By progressively reducing the search space and effectively propagating such information, it was able to efficiently compute the cost volume for high likelihood hypotheses and achieve savings in both memory and computation. All models were trained on four Nvidia-TitanXp GPUs.

In [16], dual support windows for each pixel were established, that is, a local window and the whole image window were considered at the matching process. As such, the primitive connectivity between each pixel and its neighbouring pixels in the local window are maintained, and then each pixel not only gets appropriate supports from neighbouring pixels within its local support window, but also receives more adaptive support from the other pixels outside the local window. Furthermore, a local edge-aware filter and a non-local edge-aware filter were merged to achieve collaborative filtering of the cost volume. All the experiments were conducted on a PC with a 3.2 GHz Intel Core i5-6500 CPU and 8-GB memory.

The architecture developed in [17] obtains dense stereo reconstructions using high-resolution images for infrastructure inspections. The proposed approach used a less resource-demanding non-learning method, guided by a learning-based model. This allows to handle high-resolution images and achieve accurate stereo reconstruction. First, a deep-learning model produced an initial disparity prediction with uncertainty for each pixel of the down-sampled stereo image pair. Then, a downstream process performed a modified version of a generic semi-global block matching method with the up-sampled per-pixel searching range. This approach was trained with a mini-batch of four on four NVIDIA TITAN X GPUs for five epochs.

The authors of [18] presented a local stereo matching algorithm, whose main novelty relies on the block-matching aggregation step. This is an adaptive support weights approach in which the weight distribution favours pixels that share the same displacement with the reference is presented. The
proposed weight function depended additionally on the tested shift, by giving more importance to those pixels in the block-matching with a smaller cost. The proposed approach was embedded in a pyramidal procedure to locally limit the search range, which helped to reduce ambiguities in the matching process and saved computational time. The method has been implemented on a MacOS system using a 3.3 GHz Intel Xeon CPU.

In [19], a GPU architecture for real-time semantic stereo matching was proposed. The proposed framework relied on coarse-to-fine estimations in a multi-stage fashion, allowing: 1) Very fast inference even on embedded devices, with marginal drops in accuracy. 2) Trade accuracy for speed, according to the specific application requirements. Experimental results of high-end GPUs as well as on an embedded Jetson TX2 demonstrated the superiority of semantic stereo matching compared with standalone tasks and highlight the versatility of their framework on any hardware and for any application.

In [20], a deep self-guided cost aggregation method, to obtain an accurate disparity map from a pair of stereo images was proposed. Based on observations, each cost volume slice could guide itself based on the internal features. However, finding a direct mapping function from the initial and filtered cost volume slice without any guidance image was difficult. To solve this problem, an advanced deep learning technique to perform self-guided cost aggregation was proposed. The algorithm was implemented on an Intel i7 4770 @ 3.4 GHz with 16 GB RAM.

In [21], a novel approach to binocular stereo for fast matching of high-resolution images was presented. The presented formulation built prior on the disparities by forming a triangulation on a set of support points which could be robustly matched, reducing the matching ambiguities of the remaining points. This allowed for the efficient exploitation of the disparity search space, yielding accurate dense reconstruction without the need for global optimization. Further, the proposed approach automatically determined the disparity range and could be easily parallelised. All experiments were conducted on a single i7 CPU core running at 2.66 GHz.

A new taxonomy of Recursive Edge-Aware Filters (REAF) suitable for the stereo matching problem was provided in [22]. The one-tap recursive filters were classified according to recursion rate calculation, recursion type, and the unification of reverse directions. Experimental results demonstrated the advantages of un-normalized recursion for matching accuracy and sequential integration of reverse directions for execution speed. These are important conclusions for future directions of REAFs. The experiments were conducted on i5 3.10 GHz 32-bit CPU with 4 GB RAM plus Nvidia Quadro4000 graphics card.

The authors in [23] presented an improved stereo matching algorithm which utilised per-pixel difference adjustment for Absolute Difference and gradient matching to reduce the radiometric distortions. Then, both differences were combined with census transform to reduce the effect of illumination variations. A new approach in which an iterative guided filter was introduced at a cost aggregation to preserve and improve the object boundaries. The undirected graph segmentation was used at the last stage to smoothen the low textured areas. All of the experiments were run on the central processing unit (CPU) with CORE i5 3.2 GHz and 12 GB of RAM.

In [24], a multi-stage stereo matching technique was developed to improve the matching cost computation stage where windows size of the Sum of Absolute Difference (SAD) and threshold adjustment were applied. They also used the median filter as the primary filter in the refinement disparity map stage to overcome the precision limitation of the disparity map.

Finally, in [25] a deep learning architecture for stereo disparity estimation was proposed. The proposed Atrous Multiscale Network (AMNet) adopts an efficient feature extractor with depthwise-separable convolutions and an extended cost volume that deployed stereo matching costs on the deep features. A stacked atrous multiscale network was proposed to aggregate rich multiscale contextual information from the cost volume which was allowed for estimating the disparity with high accuracy at multiple scales. All models were implemented with PyTorch and NVIDIA GPU trained.

3 | THE PROPOSED ALGORITHM

Several previous works that looked for real-time processing and embedded capabilities have used SAD (Sum of Absolute Differences) as a local metric correlation [26–28]. This is due to the fact that SAD has a regular structure with a fixed running time which facilitates hardware implementations that are suitable for real-time processing. To obtain the SAD formulation, the correlation windows were compared as shown in Equation (1): where \( I_l(x + i, y + j) \) (left image) and \( I_r(x + i + s, y + j) \) (right image) were the greyscale values of the pixels within the window in both images. The equation (2 \( w + 1 \))^2 represented the correlation window size, where \( s \) was the shift of the window in the right image and \( s_m \) was the maximal shift of the correlation window in the right image. Finally, a correlation coefficient was determined for each pixel and the shift which minimised the correlation coefficient was retained as the disparity.

\[
C_l(x, y, s) = \sum_{i=-w}^{i=w} \sum_{j=-w}^{j=w} |I_l(x + i, y + j) - I_r(x + s + i, y + j)|
\]

To improve the scope and performance of the traditional SAD-based stereo matching formulation, in this work a hypothesis was proposed stating that a disparity map that uses small correlation windows under object boundaries and large correlation windows for homogeneous areas could deliver promising results in terms of accuracy. Furthermore, the cornerstone of the proposed algorithm was a local metric correlation (SAD) and FPGA implementation was feasible and suitable for real-time processing and embedded capabilities. In
Figure 1, the block diagram of the proposed algorithm was shown; first, dense disparity maps were computed. To obtain this, the Sum of Absolute Differences (SAD) was used as a local metric correlation for different window sizes \((w_1, w_2, w_3, \ldots, w_n)\). Then, left-right consistency at each window size was estimated. Finally, Sobel and the CPD (Centre Pixel Deviation) were used as a source to estimate each disparity map; the aforementioned maps were used as inputs to a fuzzy inference system which combined the disparities in order to generate a final disparity map. This disparity map used small correlation windows under object boundaries and large correlation windows for homogeneous areas.

In order to validate the hypothesis, in Figure 2, the performance of different combinations of correlation windows sizes were shown; moreover, Table 1 showed the calculation of six different window sizes and the combination between them; as a result, the percentage of success was obtained by comparing the hypothetical, estimated disparity map with the ground-truth (which combined the different disparity maps generated for each correlation window size to minimise the estimation error). The column labelled as Window (see Table 1) represents the combinations of the different window sizes. The Row \(w\) allows for interpretation of an indicative value for the window size; where, the number one is the \(3 \times 3\) window, two of \(5 \times 5\), three has a size of \(9 \times 9\), four is \(13 \times 13\), five of \(17 \times 17\) and finally six of \(21 \times 21\). For the dataset used, it is the well-known Tsukuba scene which consists of a stereo pair with 77 maximum disparity. The error percentage is defined as the percentage of ‘bad’ pixels. A ‘bad’ pixel is defined as any pixel \((pix_{i,j})\) whose

![Figure 1](image1.png)  
**FIGURE 1**  Block diagram of the proposed algorithm

![Figure 2](image2.png)  
**FIGURE 2**  Percentage of error in individual windows and combinations of windows
absolute difference between the ground truth and the estimated disparity (groundTruth$_{i,j}$ - estimatedDisparity$_{i,j}$) is greater than a quality threshold (in this hypothesis validation threshold = 1). To select a “proper” window combination, two important criteria were considered; the first was the lower percentage of error, and the second was the fewer windows used in the process. For the combination “1 = 3 − 5”, an error equal to 7.62% was reached; other combinations were capable of improving this performance, nevertheless, more windows were used which led to the increase of the usage of more computational resources. Although all the windows were used, the obtained error (6.66%) has insignificant outperformance compared with other more efficient combinations in terms of computational requirements. Therefore, the combination ‘1 = 3 − 5’ reached the best performance (in terms of accuracy and computational requirements), as illustrated in Figure 3 and this is the configuration that is recommended when implementing the proposed algorithm.

### 3.1 Correlation computation and inference parameters

Let $SAD(x, y, d)$ be the mathematical formulation of the Sum of Absolute Differences, Equation (2); where $f$ was the value in the grayscale and the subscript $r$ and $t$ were from the reference image and the target, respectively; located in the $(x, y)$ position. $2 \times w + 1$ was the size of the correlation window, $d$ was the correlation index and the maximum shift of the correlation window in the search image is $d_{max}$ as illustrated in Equation (3). The disparity $d(x, y)$ between the one pixel of the target image and the same pixel in the reference image was defined as the displacement $d$ that minimises the correlation index. The original SAD formulation used a square fixed window (correlation window) around the pixel of interest in the target image; the window correlated with a second window that moved in all positions $d$ in the second image; The position where the lowest correlation value was obtained determined the disparity value corresponding to the pixel of interest. Three different windows sizes ($3 \times 3, 9 \times 9$ and $17 \times 17$) were used in our algorithmic formulation. Two disparity maps were calculated for each window size, one based on the correspondences of the left image with those on the right, and another based on the correspondences on the right with those on the left; then, inconsistent disparities were produced by occlusions in the scene, there was visualization in a certain region that was not seen in the other [29].

$$SAD(x, y, d) = \sum_{i=-w}^{j=w} \sum_{j=-w}^{j=w} |I_r(x + i, y + j) - I_t(x + i + d, y + j)|,$$

$$d(x, y) = \arg\min_d SAD(x, y, d), \quad (3)$$

| Windows | 1 = 3x3 | 2 = 5x5 | 3 = 9x9 | 4 = 13x13 | 5 = 17x17 | 6 = 21x21 |
|---------|---------|---------|---------|-----------|-----------|-----------|
| 1       | w       | 1       | 2       | 3         | 4         | 5         | 6         |
| %       | 26.81%  | 17.09%  | 12.73%  | 11.91%    | 12.26%    | 12.76%    |
| 2       | w = 1-2 | 1-3     | 1-4     | 1-5       | 1-6       | 2-3       | 2-4       | 2-5       | 2-6       | 3-4       |
| %       | 10.4%   | 9.58%   | 9.18%   | 8.96%     | 8.76%     | 9.84%     | 9.3%      | 9.01%     | 8.78%     | 9.88%     |
| w       | 3-5     | 3-6     | 4-5     | 4-6       | 5-6       |           |           |           |           |           |
| %       | 9.41%   | 9.1%    | 9.88%   | 9.39%     | 9.82%     |           |           |           |           |           |
| 3       | w = 1-2-3 | 1-2-4   | 1-2-5   | 1-2-6     | 1-3-4     | 1-3-5     | 1-3-6     | 1-4-5     | 1-4-6     | 1-5-6     |
| %       | 8.82%   | 8.32%   | 8.15%   | 8.14%     | 8.48%     | 7.62%     | 7.8%      | 8.33%     | 7.93%     | 8.25%     |
| w       | 2-3-4   | 2-3-5   | 2-3-6   | 2-4-5     | 2-4-6     | 2-5-6     | 3-4-5     | 3-4-6     | 3-5-6     | 4-5-6     |
| %       | 8.75%   | 8.33%   | 8.09%   | 8.48%     | 8.09%     | 8.33%     | 9.04%     | 8.64%     | 8.74%     | 9.16%     |
| 4       | w = 1-2-3-4 | 1-2-3-5 | 1-2-3-6 | 1-2-4-5   | 1-2-4-6   | 1-2-5-6   | 1-3-4-5   | 1-3-4-6   | 1-3-5-6   | 1-4-5-6   |
| %       | 7.84%   | 7.44%   | 7.2%    | 7.57%     | 7.21%     | 7.43%     | 7.74%     | 7.4%      | 7.48%     | 7.72%     |
| w       | 2-3-4-5 | 2-3-4-6 | 2-3-5-6 | 2-4-5-6   | 3-4-5-6   |           |           |           |           |           |
| %       | 8.02%   | 7.67%   | 7.75%   | 7.88%     | 8.42%     |           |           |           |           |           |
| 5       | w = 1-2-3-4-5 | 1-2-3-4-6 | 1-3-4-5-6 | 1-2-4-5-6 | 1-2-3-5-6 | 1-2-3-4-6 |           |           |           |           |
| %       | 7.15%   | 6.84%   | 6.91%   | 7.02%     | 7.21%     | 7.48%     |           |           |           |           |
| 6       | w       | All     | %       | 6.66%     |           |           |           |           |           |           |
Once the disparities for each window size were estimated, the smaller size was selected as the base. A FIS (fuzzy inference system) with two different input parameters was used to infer which correlation window size was more suitable for each pixel in the scene. Let \( I_l \) be the left image of the input stereo-par, the first input parameter for the inference process was the image gradient. The gradient of the base window \( S(X, Y) \) was calculated using the Sobel operator; where the mask \( G_x \) was determined by the Equation (4) for the horizontal borders and \( G_y \) for the vertical borders by the Equation (5) the combination of both allows to obtain the gradient per pixel Equation (6).

\[
G_x = \begin{bmatrix} -1 & 0 & 1 \\ -2 & 0 & 2 \\ -1 & 0 & 1 \end{bmatrix} \quad (4)
\]
\[
G_y = \begin{bmatrix} 1 & 2 & 1 \\ 0 & 0 & 0 \\ -1 & -2 & -1 \end{bmatrix} \quad (5)
\]
\[
Sobel = \sqrt{G_x^2 + G_y^2} \quad (6)
\]

The result obtained with the horizontal and vertical gradient masks of the Tsukuba scene is shown in Figure 4. Another input parameter for the inference process is the Centre Pixel Deviation (CPD), where \( x_{cp} \) is the reference point and the neighbours are evaluated to know the difference of the values by means of the Equation (7), that is, the Centre Pixel Deviation (CPD) is a numerical index of the dispersion of a data set in a \( 5 \times 5 \) block.

---

**FIGURE 3** The original “Tsukuba” scene. It shows disparity maps by applying the original SAD formulation. Small correlation windows needed to be used, see (a) in order to preserve edges; for homogeneous areas, only large correlation windows delivered accurate results, as illustrated in (b) and (c). When small correlation windows under object boundaries and large correlation windows for homogeneous areas were used, (d) a significant improvement was achieved.

**FIGURE 4** Gradient result of the Tsukuba scene.

---

**FIGURE 4** Gradient result of the Tsukuba scene.
\[
CPD = \sqrt{\frac{1}{N-1} \sum_{i=1}^{N} (x_i - \mu)^2}
\]  

(7)

The greater the deviation of the centre pixel, the greater the dispersion of the neighbours, where the calculated deviation was an average of the individual deviations of each observation with respect to the reference pixel, which was the one in the centre. Thus, it allows measuring the degree of dispersion or variability. First, by measuring the difference between each value in the data set and the one in the centre. Then, adding all these individual differences to give the total of all the differences. Finally, dividing the result by the total number of observations to reach an average of the distances between each individual observation and the one in the centre. This average of the distances was the Pixel Deviation from the centre and thus represented dispersion Figure 5.

The main function of CPD was to evaluate the ratio of the pixel in the process concerning its neighbours. Having a greater dispersion represented two situations: the first was that the affected area had two or more objects or was a textured area and the second was that the scene had errors in its appearance. The advantage of using CPD helped homogeneous highlighted areas by employing larger windows.

3.2 The inference process

The pixel deviation from the centre of the windows \((d_{w2}, d_{w3})\), and the detection of the edges obtained by Sobel from the left image \(d_{L(Sobel)}\) at each point \((x, y)\) for each level of disparity \(d\) were subjected to a diffuse analysis that determined which window size and its previously computed disparity was assigned in the final disparity map. For the FIS formulation, entry membership functions \(CPD_{w2}, CPD_{w3}\) were used and Sobel determined the degree of membership of the fuzzy sets \(Min, Med\) and \(Max\) for \(CPD_{w}\) and \(NoEdge, PosEdge\) and \(Edge\). In the CPD membership function for the sets \(Min\) and \(Med\), it was proposed to use triangular type functions and for \(Max\) keystone functions, where \(Min\) was an environment similar to that of the central pixel, \(Med\) had values with similar neighbours and some with a high difference and finally \(Max\) was an expression of a totally radical neighbourhood in its values.

Membership functions of fuzzy logic for the CPD parameter is shown in the Equation (8–10):

\[
\mu_{CPD_{dis}}(x; -4, 0, 4) = \begin{cases} 
0 & x \leq 0 \\
-x & 0 < x \leq -4 \\
20 - x & -4 < x \leq 4 \\
20 & x > 4
\end{cases}
\]  

(8)

\[
\mu_{CPD_{med}}(x; 0, 4, 20) = \begin{cases} 
0 & x \leq 0 \\
20 - x & 0 < x \leq 4 \\
20 & 4 < x \leq 20 \\
0 & x > 20
\end{cases}
\]  

(9)

\[
\mu_{CPD_{max}}(x; 4, 20, 250, 260) = \begin{cases} 
0 & x \leq 4 \\
20 - x & 4 < x \leq 20 \\
250 - x & 20 < x \leq 250 \\
0 & x > 250
\end{cases}
\]  

(10)

**Figure 5** Graph of the deviation of the grouped and dispersed data with reference to the centre pixel.
The input functions for the **NoEdge** and **PosEdge** sets were defined using triangular type functions; and **Edge** used a trapezoidal function; the input membership functions settings were defined as shown in Equations (11–13).

\[
\mu_{S_{\text{NoEdg}}} (x; -50, 0, 50) = \begin{cases}
0 & x \leq 0 \\
\frac{x - (0)}{50 - 0} & 0 \leq x \leq 50 \\
-50 & 50 \leq x \leq 0 
\end{cases}
\]  
(11)

\[
\mu_{S_{\text{PosEdg}}} (x; 0, 50, 100) = \begin{cases}
0 & x \leq 0 \\
\frac{x - 0}{50 - 0} & 0 \leq x \leq 50 \\
\frac{100 - x}{50 - 0} & 50 \leq x \leq 100 \\
0 & 100 \leq x 
\end{cases}
\]  
(12)

\[
\mu_{S_{\text{Edg}}} (x; 50, 100, 1200) = \begin{cases}
0 & x \leq 50 \\
\frac{x - 50}{100 - 50} & 50 \leq x \leq 100 \\
1 & 100 \leq x \leq 1100 \\
\frac{1200 - x}{1200 - 1100} & 1100 \leq x \leq 1200 \\
0 & 1200 \leq x 
\end{cases}
\]  
(13)

The fuzzy inference rules transmitted by the restrictions are those presented in Table 2:

|   | SOBEL | PossibleEdge | Edge |
|---|-------|--------------|------|
| CPD_{w2} | Low | Medium | High | Low | Medium | High | Low | Medium | High |
| CPD_{w2} | 3 | 2 | 2 | 3 | 2 | 2 | 2 | 1 | 1 | 1 | 1 |
| Abbreviation: CPD, center pixel deviation. |

**TABLE 2**  Base of fuzzy rules

**FIGURE 6**  FPGA architecture of the proposed algorithm
number which was the x-coordinate (x) of the centre of gravity of the diffuse set of output; through the Equation (14); where $\mu_y$ was the membership function of the output set $Y$, whose output variable was $y$. $S$ was the domain or range of integration.

$$Y_d = \frac{\int y \mu_y(y)dy}{\int \mu_y(y)dy} \quad (14)$$

4 | FPGA ARCHITECTURE FOR REAL-TIME PROCESSING

In Figure (6) an overview of the FPGA architecture for the proposed algorithm is shown. The architecture was centred on an FPGA implementation where all recursive/parallelisable operations were accelerated in the FPGA fabric. First, the ‘Buffer_ImL’/‘Buffer_ImR’ was used to hold local sections of the left-right (stereo pair) images that were processed and allowed for local parallel access that facilitated parallel processing. Then, ‘BufferSAD_wn’ units, where $n$ was the correlation window size, were used to hold local patches for the correlation computation. Finally, the ‘Module Sobel’ and the ‘Module_CPD’ computed in parallel the input parameters for the fuzzy inference system which estimated the corresponding disparity value. In the following paragraphs, details about the algorithm parallelisation are shown.

The Buffer_ImL consisted of input of ‘8 bits’ and 41 outputs of ‘8 bits’. The same for the Buffer_ImR. These modules aimed to transmit the complete lines of the left-right (stereo pair) after providing the values of the image in form of windows. Both modules (Buffer_ImL and Buffer_ImR) were internally a buffer17x17 which was composed of 18 RAMs as presented in Figure (7), where 17 were in reading mode and one in writing mode, which was the last position. The one in writing mode stored the new values from the input. The reading and writing modes alternated each other in the process. The first RAM1 module in writing begins, then it goes to the RAM2 module until it reaches the RAM18 modules, this change of reading and writing operation was defined by the driver17x17 module which with the help of a counter defined the duration and the RAM module that operated in reading or writing mode. When the Ram1 module to the Ram17 module process is completed in the reading mode, this meant that it presented the corresponding 17 lines (rows) of the original image; thus, the matrix17x17 module delivered the corresponding 41 window data that worked with the pixel reallocation criteria.

The BufferSAD_wn; where $n$ was the correlation window size, was responsible for calculating the disparity for the different windows. For example, in Figure 8 the FPGA architecture for the BufferSAD_w3 is shown. It consisted of three different modules implemented in parallel-pipeline form. The Buffer3x3 module was responsible for taking the 50 8-bit inputs of the previous modules and then applying the SAD correlation method, since the search range was in a disparity of

![Figure 7](image1.png)

**FIGURE 7** Diagram of a buffer17x17 module. Similar architectures, (i.e., bufferNxN), were used for the different steps of the developed architecture

![Figure 8](image2.png)

**FIGURE 8** The 3x3 module diagram to calculate the disparities by Left and Right
16, 16 outputs were present and search for correlation was from right to left and 16 more for the search from left to right with a total of 32 pins of 13-bit outputs; this process was carried out in order to realize the left-right coherence. The muxL_3x3 modules work with the correlation bits of the left-right search and with the right-left search data, respectively. Both modules worked on the principle of a 16-bit multiplexer and a 4-bit output; where only one with the lowest value can be selected and then assigned a disparity value based on the input that presented the value with the lowest correlation magnitude. Finally, the Minor_disp module was chosen as input of the two previous multiplexer modules and then it was stored in a buffer (memory) of the disparity size, and the disparity of the left image and the disparity of the right image is compared, the smaller one was chosen, obtaining the final disparity value.

The Sobel module: This module took the same eight inputs from the left image of the module SAD_size3x3, with the exception the central pixel, in which the Sobel mask was applied later in that value, it was divided into the size of the window that was 3x3, and to do this, the caseEdges module was used, where it used a case in which the division has been done previously to save logical elements in the operations. For the hardware architecture, see Figure 9. In general, the Sobel_Module stored lines of the Sobel image that was created to then generate the data of a 3x7 window of the new image, while the module mux_Sobel consisted of a multiplexer that selects the largest value.

The Module_CPD: This module calculated the deviation of the centre pixel, the disparity map of the proposed windows had three inputs of ‘4 bits’ and three outputs of ‘8 bits’, and the module consisted of two main internal units: 1 - Buffer_Rules5: such as the previous buffers, it stored the part of the image that was created with the disparity values and then delivered the image values in the form of a 5x5 window, within a module of input of ‘8 bits’ and 25 outputs of ‘8 bits’.

The DesvEstPixCent: In this module, the deviation rule for the 5x5 window with an 8-bit output was applied. One of the problems was to calculate the square root and the powers of the formula.

The module Rules_FIS: This module consisted of six inputs, three of them were of 4 bits, and the others were of 8 bits for a 4-bit output. The module acquired the resulting value of the gradient with Sobel, the three disparity values of the windows, and two of the Centre Pixel Deviation (CPD), as presented in the Figure 10. It consisted of 4 case modules, where the caseEdges module, CPDw2: case_CPDw2, and CPDw3: case_CPDw3 provided us 3-bit outputs with a value that depended on the input value for each one, where it was defined if it was high, medium, or low. The last caseRules module provided the three values that the previous modules calculated to convert it into a 9-bit vector, and once calculated, a case

**Figure 9** Diagram of inputs and outputs of the CPD and Sobel modules. CPD, center pixel deviation

**Figure 10** Diagram of the inputs and outputs of the fuzzy rules module
according to the order is entered and according to the fuzzy rules defined, one of the inputs was selected as the output: DispW1, DispW2 or DispW3 corresponding to the disparity value of the proposed windows.

5 | DISCUSSION AND ANALYSIS OF RESULTS

The developed FPGA architecture was implemented using a top-down approach. All modules were encoded in VHDL and simulated using post-synthesis simulations performed in ModelSim SE-64 10.5 to verify their functionality. Quartus Prime Lite Edition 18.0 was used for the synthesis process and the implementation of FPGA. A Cyclone V 5CSE-BA6U23I7 FPGA integrated into an Altera DE10-Nano development board was used. The consumption of hardware resources for the developed FPGA architecture is shown in Table 3.

The Table 4 shows quantitative results for the proposed algorithm compared with the current state of the art. To show this, we have submitted our results to the ‘Middlebury Stereo Evaluation - Version 3’. The ‘Middlebury Stereo Evaluation v.3’ is a popular dataset for stereo matching evaluation. The datasets are split into test and training sets with 15 image pairs each. Both datasets have public tables listing the results of all submitted methods. Ground-truth disparities are only provided for the training set. To have a proper comparison baseline, the error margins, ‘Average (Bad 2.0)’, were obtained from the Middlebury webpage and considering the test images (without public ground truth) of the dataset. For the error metric, global accuracy can be defined as the percentage of ‘bad’ pixels; a ‘bad’ pixel is defined as any pixel \((pix_{ij})\) whose absolute difference between the ground truth and the estimated disparity \((\text{groundTruth}_{ij} - \text{estimatedDisparity}_{ij})\) is greater than a quality threshold “bad 2.0”.

The analysis of Table 4 after comparing the results with previous works demonstrated relatively high performance in terms of accuracy and high processing speed; the percentage of error obtained was 30.9% and the processing speed was 0.03 s (for more details please see http://vision.middlebury.edu/stereo/eval3/and look for the ‘MANE’ algorithm). For the leader table, the proposed approach was ranked in the 84th place, however, most algorithms in the leader table were based on advanced stereo matching techniques and this limits the processing speed. As a result, the most accurate algorithms [30] (error 5.1%), [31] (5.43%), had a poor performance in terms of processing speed, 103th and 107th, respectively. Besides, most accurate algorithms in the leader table required high-end processors such as i7-7700K or i7-4770K and this limited the embedded capabilities.

For previous algorithms with similar accuracy, the most accurate approach was the ELAS_ROB (2010) [21] algorithm with an error of 27.3%. Other works, such as DAWA-F (2019) [18] and SGBMP (2019) [17] delivered similar performance in terms of accuracy, an error of 27.4% and 27.8%, respectively. But in all cases, the proposed approach obtained an error of 30.9%, it outperformed these previous works in terms of processing speed (this approach ranked sixth place of the Middlebury leader table in terms of processing speed), reaching 0.03 s of processing time compared with previous works; 0.48 s for ELAS_ROB (2010) [21], 1683 s for DAWA-F (2019) [18] and 9.12 s for SGBMP (2019) [17], see Table 5. Other previous works such as REAF (2015) [22] with the error of 33.8%

| Method          | Average (Bad 2.0) | Running Environment       |
|-----------------|-------------------|---------------------------|
| ELAS_ROB (2010) | 27.3              | Intel core i7-6700K       |
| DAWA-F (2019)   | 27.4              | Intel Xeon 6-Core@3.33 GHz|
| SGBMP (2019)    | 27.8              | Nvidia Titan X            |
| FASW (2019)     | 28.6              | Intel core i5-6500@3.2GHz |
| DeepPruner_ROB (2019) | 30.1       | GPU @ 2.5 Gz              |
| MSMD_ROB (2018) | 30.9              | Nvidia Titan X + 8 cores  |
| MANE (proposed) | 30.9              | Cyclone V 5CSEBA6U23I7 FPGA|
| REAF (2015)     | 33.8              | Intel core i5@3.10GHz     |
| DSGCA (2018)    | 27.3              | i7-4770@3.40 GHz; GTX 1080 GPU|
| IGF (2017)      | 34.0              | i5@3.2 GHz; 12 GB of RAM  |
| SM-AWP (2019)   | 38.1              | Intel core i7@2.7 GHz     |
| AMNet (2019)    | 53.3              | Nvidia Titan XP           |
processing time of 6.87 s; IGF (2017) [23] with an error of 34.0%, the processing time of 132 s; or SM-AWP (2019) [24] with an error of 38.1, processing speed of 1.21 s, was outperformed in both accuracy and speed processing by the proposed algorithm.

In the other test, the results were submitted to the ‘ETH3D Multi-view stereo/3D reconstruction dataset’ proposed by Schöps et al. [32]. This dataset covered a variety of indoor and outdoor scenes. Ground-truth geometry has been obtained using a high-precision laser scanner. A DSLR camera as well as a synchronized multi-camera rig with varying field-of-view was used to capture images. For the two-view stereo challenge, 27 training and 20 test frames (data/results)‐view stereo on frames of the multi-camera rig were provided. To have a proper comparison baseline, the error margins, ‘Average (Bad 0.5)’, were obtained from the ETH3D webpage and considering the test images (without public ground truth) of the dataset. For the error metric, the global accuracy can be defined as the percentage of ‘bad’ pixels; a ‘bad’ pixel is defined as any pixel (\( \text{pix}_{ij} \)) whose absolute difference between the ground truth and the estimated disparity (\( \text{groundTruth}_{ij} - \text{estimatedDisparity}_{ij} \)) is greater than a quality threshold ‘bad 1.0’. Quantitative results for the proposed algorithm compared with the current state of the art were presented in Table 6. Qualitative results demonstrated relatively high performance in terms of accuracy and high processing speed; the percentage of error obtained was 23.11% (for more details please see https://www.eth3d.net/low_res_two_view?metric = bad-0.5 and look for the “MANE” algorithm). For previous algorithms with similar accuracy, the most accurate approach is the GANetREF_R VC [33] algorithm with an error of 25.41%. Other works, such as PASM [34], LSM [35] and DispFullNet [36] delivered similar performance in terms of accuracy, error of 28.19%, 29.98% and 30.27%, respectively. Other previous works such as, ELAS [21] with error of 33.68%, ELAS_RVC [21] with error of 33.79%, NVStereoNet_ROB [37] with error of 41.93%, the combination of SGM + DAISY [38,39] with error of 54.67, SPS-STEREO [40] with error of 55.62 or MFMNet_re [41] with error of 72.05% are outperformed in terms of accuracy.

With regard to the previous hardware implementations, the proposed approach outperformed previous FPGA-based approaches in terms of processing speed reaching 427\@640 \times 480 compared with the previous works; 427\@640 \times 480 for Perri et al. [42] (2013) and 68\@640 \times 480 for Hummenberger et al. [43], this outperformed the proposed approach, but with an \( \times 3 \) clock speed. In both cases [42] (2013), [43] a similar disparity range (15) was considered, please see Table 7. Other previous studies such as Zha et al. [44] with a processing speed of 30\@1920 \times 1680 or Ttofis et al. [45] with a processing speed of 60\@1280 \times 720, process higher image resolution compared with the proposed approach, however, these researches used a higher clock speed and this could have limited the small embedded capabilities.

### Table 5 The Average of the total time

| Method                  | Average (Total Time) |
|-------------------------|----------------------|
| MANE (proposed)         | 0.03                 |
| DeepPruner_ROB (2019)   | 0.13                 |
| AMNet (2019)            | 0.40                 |
| ELAS_ROB (2010)         | 0.48                 |
| MSMD_ROB (2018)         | 0.79                 |
| SM-AWP (2019)           | 1.21                 |
| REAF (2015)             | 6.87                 |
| SGBMP (2019)            | 9.12                 |
| DSGCA (2018)            | 11.0                 |
| FASW (2019)             | 67.5                 |
| IGF (2017)              | 132.0                |
| DAWA-F (2019)[18]       | 1683.0               |

### Table 6 The average disparity errors on test dense (bad = 0.5), dataset two-view stereo of ETH3D

| Method                  | Average (bad 0.5) | Running environment |
|-------------------------|-------------------|---------------------|
| GANetREF_RVC (2019)     | 25.41%            | Intel i9-9920X,     |
|                         |                   | Nvidia Quadro RTX 8000 |
| PASM (2019)             | 28.19%            | RTX 2080Ti          |
| LSM (2020)              | 29.98%            | TITAN X GPU         |
| DispFullNet (2018)      | 30.27%            | GTX 1080Ti0ti       |
| MANE (proposed)         | 31.97%            | Cyclone V 5CSEBA6U23I7 FPGA |
| ELAS (2010)             | 33.68%            | Intel core i7-6700K |
| ELAS_RVC (2010)         | 33.79%            | Intel core i7-6700K |
| NVStereoNet_ROB (2018)  | 41.93%            | Titan XP GPU        |
| SGM + DAISY (2018)      | 54.67%            | Intel core i7-6700K |
| SPS-STEREO (2014)       | 55.62%            | Intel core i7-6700K |
| MFMNet_re (2019)        | 72.05%            | Nvidia GeForce 1080Ti |

### Table 7 Running environment

| Method                  | Average (bad 0.5) | Running environment |
|-------------------------|-------------------|---------------------|
| GANetREF_RVC (2019)     | 25.41%            | Intel i9-9920X,     |
|                         |                   | Nvidia Quadro RTX 8000 |
| PASM (2019)             | 28.19%            | RTX 2080Ti          |
| LSM (2020)              | 29.98%            | TITAN X GPU         |
| DispFullNet (2018)      | 30.27%            | GTX 1080Ti0ti       |
| MANE (proposed)         | 31.97%            | Cyclone V 5CSEBA6U23I7 FPGA |
| ELAS (2010)             | 33.68%            | Intel core i7-6700K |
| ELAS_RVC (2010)         | 33.79%            | Intel core i7-6700K |
| NVStereoNet_ROB (2018)  | 41.93%            | Titan XP GPU        |
| SGM + DAISY (2018)      | 54.67%            | Intel core i7-6700K |
| SPS-STEREO (2014)       | 55.62%            | Intel core i7-6700K |
| MFMNet_re (2019)        | 72.05%            | Nvidia GeForce 1080Ti |
Concerning the computer hardware requirements, low hardware requirements were required to perform the different experiments presented in this academic research (a CycloneV 5CSEBA6U2317 FPGA integrated into an Altera DE10-Nano development board). Comparing this system’s hardware and software with previous works (Table 4) which used high end processors such as Intel Xeon 6-Core@3.33 GHz for DAWA-F (2019) [18], Nvidia Titan X for SGBMP (2019) [17] or Nvidia Titan XP for AMNet (2019) [25]; it was concluded that this approach allowed an efficient implementation into embedded processors (making it possible to develop a smart camera/sensor suitable for embedded applications). Finally, for qualitative results in Figures 11–16 the estimated disparity maps for the test images (without public ground truth) of the

| Method | Implementation Platform | Disparity Range | FPS | Clock |
|--------|-------------------------|-----------------|-----|-------|
| Aguilar-Gonzalez et al. (2016) [46] | FPGA Altera Cyclone IV | NA | 117@1280 × 720 | NA |
| Zha et al. (2016) [44] | FPGA Xilinx Kintex-7 | 60 | 30@1920 × 1680 | 160 MHz |
| Werner et al. (2014) [47] | NA | 60 | 30@1920 × 1680 | 180 MHz |
| Ttofis et al. (2014) [45] | FPGA Xilinx Kintex-7 | 64 | 60@1280 × 720 | 109 MHz |
| Perri et al. (2013) [42] | FPGA Virtex6 XCoVLX760 | 15 | 68@640 × 480 | 120 MHz |
| Humenberger et al. (2010) [43] | GeForce GTX 280 | 15 | 573.7@320 × 240 | 3 GHz |
| Zhang et al. (2011) [48] | FPGA Altera EP3SL150 | 64 | 60@1024 × 768 | NA |
| Pérez et al. (2016) [49] | FPGA Altera EP2C35F672C6 | 15 | 76@1280 × 1024 | 100 MHz |
| Larabi et al. (2013) [50] | FPGA Xilinx Vertex-4 | 59 | 110@450 × 375 | 142 MHz |
| Banz et al. (2010) [51] | FPGA Xilinx Vertex-5 | 64 | 30@640 × 480 | 108 MHz |
| Han et al. (2016) [52] | NA | 60 | 30@640 × 480 | NA |
| Mattocia et al. (2015) [53] | FPGA Xilinx Spartan-6 | 32 | 30@640 × 480 | NA |
| MANE (proposed) | FPGA Altera Cyclone V | 16 | 427@640 × 480 | 50 MHz |
| MANE (proposed) | FPGA Altera Cyclone V | 59 | 279@640 × 480 | 50 MHz |
| MANE (proposed) | FPGA Altera Cyclone V | 77 | 155@640 × 480 | 50 MHz |

**Table 7** Stereo matching implementation FPS

![Figure 11](image1.png) Different scenes from Middlebury Stereo Evaluation - Version three

![Figure 12](image2.png) Disparity map of the proposed algorithm in different scenes from Middlebury Stereo Evaluation - Version three
Middlebury and ETH3D were presented. In all cases, accurate results with similar error compared with several works in the current literature were obtained.

6 | CONCLUSIONS

In this research article, a novel real-time stereo matching algorithm embedded in an FPGA was proposed. To address the SAD correlation window size problem, disparity maps with different window sizes were estimated. Then, the left-right consistency for each window size was computed. Finally, the centre pixel deviation and the Sobel gradient were estimated and used as a basis of a FIS (fuzzy inference system) which combined the previously computed disparities in order to generate a final disparity map that used small correlation windows under object boundaries and large correlation windows for homogeneous areas. To guarantee embedded capabilities, an FPGA architecture was proposed and experimental results demonstrated that our algorithmic formulation provided promising results (in terms of accuracy, processing speed, and hardware requirements) compared with the current state of the art.
ACKNOWLEDGEMENTS
This research was supported by CONACyT. We thank our colleagues from Tecnológico Nacional de México/IT de Tuxtla Gutiérrez who provided insight and expertise that greatly assisted this research, even though they may not agree with all of the interpretations of this paper, their contributions were really helpful.

ORCID
Héctor-Daniel Vázquez-Delgado @ https://orcid.org/0000-0002-1260-4210
Maidín Pérez-Patricio @ https://orcid.org/0000-0003-3414-3379
Abiel Aguilar-González @ https://orcid.org/0000-0002-4941-9967
Miguel-Octavio Arias-Estrada @ https://orcid.org/0000-0002-8000-9501
Jorge Luis Canas-Anzueto @ https://orcid.org/0000-0002-0379-7073

REFERENCES
1. Wang, Y., et al.: Anytime stereo image depth estimation on mobile devices. In: 2019 International Conference on Robotics and Automation (ICRA), 5893–5900. IEEE (2019)
2. Tang, F., Li, H., Wu, Y.: Fmd stereo slam: fusing mvg and direct formulation towards accurate and fast stereo slam. In: 2019 International Conference on Robotics and Automation (ICRA), pp. 133–139. IEEE (2019)
3. Cambuim, L.F., et al.: An fpga-based real-time occlusion robust stereo vision system using semi-global matching. J. Real-Time Image Process. 17, 1447–1468 (2019). doi:10.1007/s11554-019-00902-w
4. Cui, H., Dahnoun, N.: Real-time stereo vision implementation on nvidia jetson tx2. In: 2019 8th Mediterranean Conference on Embedded Computing (MECO), 1–5. IEEE (2019)
5. Faugeras, O., et al.: Real-time correlation-based stereo: algorithm, implementations and applications, pp. 1–23. INRIA, Sophia-Antipolis (1993). https://hal.inria.fr/inria-00074658
6. Kanade, T., Okutomi, M.: A stereo matching algorithm with an adaptive window: theory and experiment. IEEE Trans. Pattern Anal. Mach. Intell. 16(9), 920–932 (1994)
7. Liu, X., Chen, F., Ha, Y.: Area efficient box filter acceleration by parallelising with optimised adder tree. In: 2019 IEEE Computer Society Annual Symposium on VLSI (ISVLSI), 55–60. IEEE (2019)
8. Trofín, C., Kyrklou, C., Theoscharides, T.: A low-cost real-time embedded stereo vision system for accurate disparity estimation based on guided image filtering. IEEE Trans. Comput. 65(9), 2678–2693 (2015)
9. Gupta, H., Antony, D.S., Rathna, G.: Implementation of Gaussian and box kernel based approximation of bilateral filter using opencl. In: 2015 International Conference on Digital Image Computing Techniques and Applications (DICTA), 1–5. IEEE (2015)
10. Wang, S., Maruyama, T.: An implementation method of the box filter on fpga. In: 2016 26th International Conference On Field Programmable Logic and Applications (FPL), 1–8. IEEE (2016)
11. Cardenas, P.-F., et al.: A proposal for a soc fpga-based image processing in rgb-d sensors for robotics applications. In: 2018 IEEE 2nd Colombian Conference on Robotics and Automation (CCRA), 1–6. IEEE (2018)
12. Lee, Y., et al.: Fpga design and implementation of accelerated stereo matching for obstacle detection. In: 2019 International Conference on Electronics, Information, and Communication (ICEIC), 1–2. IEEE (2019)
13. Fathi, M., Sheikhaei, S., Tavakoli, J.: Low-cost and real-time hardware implementation of stereo vision system on fpga. In: 2019 27th Iranian Conference on Electrical Engineering (ICEE), 258–263. IEEE (2019)
14. Lu, H., et al.: Cascaded multi-scale and multi-dimension convolutional neural network for stereo matching. In: 2018 IEEE Visual Communications and Image Processing (VCIP), 1–4. IEEE (2018)
15. Duggal, S., et al.: DeepPruner: learning efficient stereo matching via differentiable patchMatch. In: Proceedings of the IEEE International Conference on Computer Vision, 4384–4393 (2019)
16. Wu, W., et al.: Stereo matching with fusing adaptive support weights. IEEE Access. 7, 61960–61974 (2019)
17. Hu, Y., Zhen, W., Scherer, S.: Deep-Learning Assisted High-Resolution Binocular Stereo Depth Reconstruction. 2020 IEEE International Conference on Robotics and Automation (ICRA). pp. 8637–8643 (2020)
18. Navarro, J., Buades, A.: Semi-dense and robust image registration by shift adapted weighted aggregation and variational completion. Image Vis. Comput. 89, 258–275 (2019)
19. Dovsi, P.L., et al.: Real-time semantic stereo matching. arXiv preprint (2019). arXiv:1910.00541
20. Park, I.K.: Deep self-guided cost aggregation for stereo matching. Pattern Recognit. Lett. 112, 168–175 (2018)
21. Geiger, A., Roser, M., Urtasun, R.: Efficient large-scale stereo matching. In: Asian Conference on Computer Vision, 25–38. Springer, Berlin (2010)
22. Cigla, C.: Recursive edge-aware filters for stereo matching. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops, 27–34 (2015)
23. Hamzah, R.A., Ibrahim, H., Hassan, A.H.A.: Stereo matching algorithm based on per pixel difference adjustment, iterative guided filter and graph segmentation. J. Vis. Commun. Image Represent. 42, 145–160 (2017)
24. Razak, S.S.A., Othman, M., Kadmin, A.: The effect of adaptive weighted bilateral filter on stereo matching algorithm. Int. J. Eng. Adv. Technol. 8, C5839028319 (2019)
25. Du, X., El-Khamy, M., Lee, J.: AMNet: Deep atrous multiscale stereo disparity estimation networks. (2019). arXiv:1904.09099. https://arxiv.org/abs/1904.09099
26. Hamzah, R., et al.: Disparity map algorithm based on edge preserving filter for stereo video processing. J Telecommun Electron Comput Eng. 10(1–7), 59–62 (2018)
27. Ivanavičius, A., et al.: Real-time cuda-based stereo matching using cyclops2 algorithm. EURASIP J. Image Video Process. 2018(1), 12 (2018)
28. Kumm, M., Kleinlein, M., Zipf, P.: Efficient sum of absolute difference computation on fpgas. In: 2016 26th International Conference on Field Programmable Logic and Applications (FPL), 1–4. IEEE (2016)
29. Chang, C., Chatterjee, S., Kube, P.R.: On an analysis of static occlusion in stereo vision. In: Proceedings of 1991 IEEE Computer Society Conference on Computer Vision and Pattern Recognition, 722–723. IEEE Comput. Soc. (1991)
30. Li, J., Ji, P., Liu, X.: Superpixel alpha-expansion and normal adjustment for stereo matching. In: Proceeding of CAD/Graphics (2019)
31. Tanai, T., et al.: Continuous 3D label stereo matching using local expansion moves. IEEE Trans. Pattern Anal. Mach. Intell. 40(11), 2725–2739 (2017)
32. Schöps, T., et al.: A multi-view stereo Benchmark with high-resolution images and multi-camera Videos. In: Conference on Computer Vision and Pattern Recognition (CVPR) (2017)
33. Zhang, F., et al.: Ga-net: guided aggregation net for end-to-end stereo matching. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 185–194 (2019)
34. Wang, L., et al.: Learning parallax attention for stereo image super-resolution. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 12250–12259 (2019)
35. Tang, C., Yuan, L., Tan, P.: LSm: learning subspace minimisation for low-level vision. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, 6235–6246 (2020)
36. Xiao, R., Sun, W., Yang, C.: Confidence inference for Focused learning in stereo matching. (2018). arXiv:1809.09758. https://deepai.org/publication/confidence-inference-for-focused-learning-in-stereo-matching.
37. Smolyanskiy, N., Kamenev, A., Birchfield, S.: On the importance of stereo for accurate depth estimation: an efficient semi-supervised deep neural network approach. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops, 1007–1015 (2018)

38. Hirschmüller, H., Scharstein, D.: Evaluation of cost functions for stereo matching. In: 2007 IEEE Conference on Computer Vision and Pattern Recognition, 1–8. IEEE (2007)

39. Tola, E., Lepetit, V., Fua, P.: Daisy: an efficient dense descriptor applied to wide-baseline stereo. IEEE Trans. Pattern Anal. Mach. Intell. 32(5), 815–830 (2009)

40. Yamaguchi, K., McAllester, D., Urtasun, R.: Efficient joint segmentation, occlusion labelling, stereo and flow estimation. In: European Conference on Computer Vision, 756–771. Springer, Cham (2014)

41. Wang, Q., et al.: Multi-path feature mining network for stereo matching. In: 2019 14th IEEE Conference on Industrial Electronics and Applications (ICIEA), 222–227. IEEE (2019)

42. Perri, S., Corsonello, P., Cocorullo, G.: Adaptive census transform: a novel hardware-oriented stereovision algorithm. Comput. Vis. Image Understand. 117(1), 29–41 (2013)

43. Humenberger, M., et al.: A fast stereo matching algorithm suitable for embedded real-time systems. Comput. Vis. Image Understand. 114(11), 1180–1202 (2010)

44. Zha, D., Jin, X., Xiang, T.: A real-time global stereo-matching on fpga. Microprocess Microsystems. 47, 419–428 (2016)

45. Tsofis, C., Theocharides, T.: High-quality real-time hardware stereo matching based on guided image filtering. In: Proceedings of the Conference on Design, Automation & Test in Europe, p. 356. European Design and Automation Association (2014)

46. Aguilar-González, A., Arias-Estrada, M.: An fpga stereo matching processor based on the sum of hamming distances. In: International Symposium on Applied Reconfigurable Computing, 66–77. Springer (2016)

47. Werner, M., Stabernack, B., Riechert, C.: Hardware implementation of a full hd real-time disparity estimation algorithm. IEEE Trans. Consum. Electron. 60(1), 66–73 (2014)

48. Zhang, L., et al.: Real-time high-definition stereo matching on fpga. In: Proceedings of the 19th ACM/SIGDA International Symposium on Field Programmable Gate Arrays, Monterey, pp. 55–64. ACM (2011)

49. Pérez-Patricio, M., et al.: An fpga stereo matching unit based on fuzzy logic. Microprocess Microsyst. 42, 87–99 (2016)

50. Larabi, S., Baha, N., Touzene, H.: Fpga implementation for stereo matching algorithm. In: 2013 Science and Information Conference, 448–454. IEEE (2013)

51. Bunz, C., et al.: Real-time stereo vision system using semi-global matching disparity estimation: architecture and fpga-implementation. In: 2010 International Conference on Embedded Computer Systems: Architectures, Modelling And Simulation, 93–101. IEEE (2010)

52. Han, J., et al.: Fpga implementation for binocular stereo matching algorithm based on sobel operator. Int. J. Database Theory Appl. 9(4), 221–230 (2016)

53. Mattoccia, S., Poggi, M.: A passive rgbd sensor for accurate and real-time depth sensing self-contained into an fpga. In: Proceedings of the 9th International Conference on Distributed Smart Cameras, 146–151. ACM (2015)