International Journal of Computational Vision and Biomechanics
Vol. 6 No. 1 (January-June, 2020)

FPGA Implementation of a Real-time Disparity Map Estimation

N. Baha and S. Larabi

Computer Science Department, University of Science and Technology – Houari Boumediene, Algiers, Algeria

Received: 14th October 2019 Revised: 17th December 2019 Accepted: 21st January 2020

In this paper, we present a new method for real-time disparity map computation based on the neural network and DSI (Disparity Space Image) data structure from a pair of stereo images. Our approach divides the disparity map computing into two main steps. The first one deals with computing the initial disparity map, using a combination of the neuronal network and the DSI structure. Whereas, the second one presents a simple and fast method to refine the initial disparity map.Three contributions are introduced so that an accurate and fast result will be reached. The first one concerns the proposition of a new strategy in order to optimize the computation time of the initial disparity map. In the second one, a specific treatment is proposed in order to obtain more accurate disparity for the neighbouring pixels to boundaries. The third one concerns the pixel similarity measure for matching score computation which consists to use in addition to the traditional pixel intensities, the magnitude and orientation of the gradients providing more accuracy. Previous methods reported on the literature are mainly restricted to implement the SAD algorithm (Sum of Absolute Differences) on FPGA (Field programmable gate Arrays). Although, the implementation presents a relative high execution time, it is known for its semi-dense disparity map. To enable both accuracy and fast real-time stereo vision embedded system, we propose a hardware parallel-pipelined implementation of our method aimed at maximizing the speed-accuracy trade-off. Experimental results using several Middlebury data sets were conducted for evaluating the proposed method.

Keywords: *FPGA, DSI (Disparity Space Image), Neural network, Occlusion, Real-time.*

1. INTRODUCTION

Stereo matching is one of the most complex tasks in machine vision. It consists in finding for each point in the left image, its corresponding in the right one. The difference between horizontal distances of these points is the disparity. A disparity map consists of all the possible disparity values in an image. Such a map is basically a representation of the depth of the perceived scene. Therefore, the disparity maps have been used to address efficiently problems such as 3D reconstruction, positioning, mobile robot navigation, obstacle avoidance. Therefore, their fast and accurate computation is still an active and incessant research topic. A great number of approaches for disparity map estimation have been proposed in the literature, including local methods [5, 6, 8, 18, 19, 21, 28] and global methods [4, 16, 17, 36, 38, 41]. A survey of the different approaches can be found in [26, 33]. Local algorithms which are based on a correlation criterion can have very efficient implementations that are suitable for real-time applications. One of the principal factors which influences the success of local methods is the proper selection of window shape and size. The windows must be large enough to capture intensity variation for reliable matching but small enough to avoid the effects of projective distortions at the same time. An appropriate window selection should improve matching accuracy but require an optimized balance between the above opposite criteria [14]. Global approaches minimize an overall cost function that involves all the pixels of the image. In these methods, calculating the disparity field is led to minimize the objective function of energy. Several optimization methods have been proposed such as dynamic programming [16], graph cuts [17], directed anisotropic diffusion [4], belief propagation [36], [41] and neural network based approaches [38]. The global methods can generate high-quality disparity maps. However, these methods are often computationally expensive and involve difficult parameter adjustment procedures that require a lot of effort to find the optimal ones, making them unsuitable for most interactive applications. Also, there are many other methods that are not strictly included in any of these two broad classes. A survey for the different approaches can be found in [26], [33].

Real-time requirements of most robot applications complicate the realization of such vision systems. The key to succeed in realizing a reliable embedded real-time-capable stereo vision system is the careful design of the core algorithm. The trade-off between execution time and quality of the matching is a difficult task and must be handled with care.

However, for extracting dense and reliable 3D information from the observed scene, stereo matching algorithms are computationally intensive. The real-time required for such application means that a task has to be finished within an a priori defined time frame. To enable both accurate and fast real-time stereo vision in embedded

Corresponding author: *E-mail: nbahatouzene,slarabi@usthb.dz*

systems, we propose a novel method for computing a dense disparity map based on the combination of Artificial Neural Network and the DSI data structure. The goal is to combine the advantages of the neural network and the DSI structure. Our approach divides the matching process into two steps: initial disparity map and refinement of the initial disparity map. Initial disparity map is first approximated by neuronal-DSI method so called (Neural-DSI). Then a refinement method is applied to the initial disparity so that an accurate result can be achieved.

Due to the computational complexity of many stereo algorithms, a number of attempts have been made to implement such systems using reconfigurable hardware in the form of FPGAs (Field programmable gate Arrays [1, 20, 23]. These devices consist of programmable logic gates and routing which can be re-configured to implement practically any hardware function. On the one hand, hardware implementations allow the application of the parallelism that is common in image processing and vision algorithms, and on the other hand the building of systems to perform specific calculations quickly compared to software implementations [12]. The use of FPGAs is now the most suitable and practical choice for hardware development. They are cheap and perform extremely well [26]. The variety of available tools make the prototyping times very short. Another advantage is that the resulting hardware implementation is open for further upgrades. Thus, FPGA implementations are very flexible and fault tolerant. The ASIC (Application Specific Integrated Circuit) implementation is an option as well, but it is more expensive, except the case of massive production. The prototyping times are considerable longer and the result is highly process-dependent. Any further changes would be difficult and can cause additionally time and money consumption increase. Their performance supremacy does in most cases not justify the choice of the ASICs. All these reasons make FPGA implementations more attractive.

Many publications [1, 9, 15, 27, 40, 43] about computing disparity maps on FPGA reported on literature use the Sum of Absolute Differences (SAD) algorithm. This choice was made to increase speed. Indeed, the SAD can be directly and easily implemented in hardware due to its simplicity. Moreover, it must be noted that the SAD algorithm results to semi-dense disparity map estimation.

The main contributions of this work are:

- The proposition of a robust matching cost based on the combination of the neural network and the DSI structure,
- The extension of matching primitives from pixel intensity to intensity, gradient magnitude and orientation of gradient vector of pixel,
- A specific treatment is proposed in order to obtain more accurate disparity for the neighbouring pixels to boundaries as will described in section 2.2.2.
- A proposition of hardware implementation of our method on FPGAs: field programmable gate array.

This paper is organized as follows: section 2 presents the stages followed to compute the disparity map. Section 3 presents the FPGA implementation of our approach. Experimental results obtained on real images are presented and discussed in section 4. Finally, section 5 concludes the paper with some remarks.

2. PROPOSED APPROACH

In our work, we assume that the images pairs are rectified. Thus the search for correspondences in the images can be limited to one dimension, ensuring a fast implementation of the stereo matching algorithm. We propose in this section the steps allowing the computation of the initial disparity map using the combination of neural network and disparity space image (DSI).

2.1. Initial Disparity Map Estimation: Neural-DSI

2.1.1. DSI Computation

Disparity Space Image (DSI) is an explicit representation of the matching space introduced by Bobik and Intille [6]. It plays an essential role in the development of the overall matching algorithm which uses the occlusion constraints. Thus, it has the advantage of improving disparities in occluded areas.

Assuming that images pairs are rectified, the search for correspondence of each feature in one image will be done in the same horizontal line of the other image. Thus the disparity computation concerns two matched points which have the same ordinate. For each pixel $p_l(x_l, y_l)$ in the left image (reference image), the disparity computation will concern all pixels of a window W_l centred on p_l where its size was experimentally chosen to be 7x7. At each pixel $p_i(x_i, y_i)$ of the W_l, the matched pixel $p_j(x_j, y_j)$ will appertain to the window W^d_r of the right image centred on $p_r(x_r, y_r)$ (see figure 1). The position of W^d_r depends on the disparity **d** associated to the pair (p_l, p_r) which varies from zero to d_{max}, where d_{max} represents the highest disparity value of the stereoscopic images (disparity range). We have thus: $x_j = x_i + s_* d$, $y_j = y_i$, where s = {+1,-1} is a sign chosen so that all disparities are positives. To determine the disparity of a given pixel $p_l(x_l, y_l)$, we calculate for each disparity **d** the score $DSI^d(p_i)$ of all pixels p_i of the windows W_l.

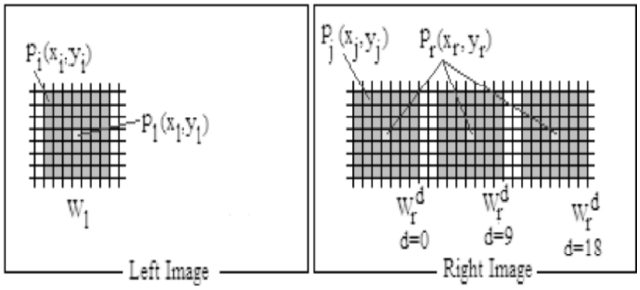

Figure 1: Different Windows used for DSI Computation

Twardowski *et al.* have presented in [37] a survey of the gradient comparison measures reported in the literature given by the following equations:

$$E = |w_{xl} - w_{xr}| + |w_{yl} - w_{yr}| \quad (1)$$

$$E = (w_{xl} - w_{xr})^2 + (w_{yl} - w_{yr})^2 \quad (2)$$

Where w_{ab}, means *a* coordinate of gradient vector *b*.

$$E = \frac{|w_l| + |w_r|}{2} - \alpha |w_l - w_r| \quad (3)$$

Where $|w_a|$ is the module of vector *a* and α is the weight parameter.

They also proposed a new matching measure using two coordinates (*m*, *x*) of the gradient vectors expressed in the following form:

$$E = |m_r - \cos(x_r - x_l)\, m_l| + \alpha |\sin(x_r - x_l)\, m_l| \quad (4)$$

Where m_r, m_l are modules of right and left gradient vectors, x_r, x_l are angles of right and left gradient vectors and α is a weight parameter. Twardowski *et al.* have compared these measures [37], the disadvantage of the measure (4) is the computation complexity relative to other measures.

In our work, to compute the matching score between two pixels $p_l(x_l, y_l)$ and $p_r(x_r, y_r)$, three attributes of pixel are used in the comparison: intensity, module and orientation of the gradient. Rather than to use only intensity, these additional attributes increase the robustness to the matching measure.

To make the algorithm more suitable for hardware-based implementations, the 3x3 Sobel [11] operator is used to compute the gradient values in *x* and *y* directions by applying the following two masks for each pixel:

$$G_x = \begin{pmatrix} 1 & 2 & 1 \\ 0 & 0 & 0 \\ -1 & -2 & -1 \end{pmatrix} \quad G_y = \begin{pmatrix} -1 & 0 & 1 \\ -2 & 0 & 2 \\ -1 & 0 & 1 \end{pmatrix} \quad (5)$$

The module is defined as:

$$|G| = |G_x| + |G_y| \quad (6)$$

The orientation $O(x, y)$ of the gradient vector is:

$$O = \tan^{-1}(G_x/G_x) \quad (7)$$

For a given disparity **d**, the score $DSI^d(p_i)$ score is computed as the sum of absolute difference of three attributes (intensity, gradient magnitude and gradient orientation) as follow:

$$DSI^d_I(p_i) = |(I_l(x_i, y_i) - I_r(x_i + s_* d, y_i))_* n_1| \quad (8)$$

$$DSI^d_G(p_i) = |(G_l(x_i, y_i) - G_r(x_i + s_* d, y_i))_* n_1)| \quad (9)$$

$$DSI^d_O(p_i) = |(O_l(x_i, y_i) - O_r(x_i + s_* d, y_i))_* n_1)| \quad (10)$$

$$DSI^d(p_i) = (DSI^d_I(p_i) + DSI^d_G(p_i) + DSI^d_O(p_i))_* n_2 \quad (11)$$

Where ($\mathbf{I_l}$, $\mathbf{I_r}$), ($\mathbf{G_l}$,$\mathbf{G_r}$), ($\mathbf{O_l}$,$\mathbf{O_r}$) are respectively the intensities, gradient magnitudes, gradient orientations values of the pixels on the left and right images, $\mathbf{n_1}$,$\mathbf{n_2}$ are the input weights of the neural network, they will be computed in the step of learning of the neural network as will be presented in subsection 3.3.

To determine the disparity of a given pixel $p_l(x_l,y_l)$, we calculate for all pixels p_i of the windows W_l the score $DSI^d(p_i)$ for different values of **d** from zero until $\mathbf{d_{max}}$.

The initial disparity of the pixel p_l is chosen as the disparity *d** with the minimal cost $DSI^{d*}(p_i)$ among the various costs of neighbouring pixels to p_l of the window W_l and will be noted $d^*(p_l)$.

As the implementation of this method for *DSI* computation is time consuming, we propose in the next section neural network architecture in order to parallelize the calculation of various costs.

2.1.2. Neural Network Architecture

The ANN (Artificial Neural Network) is a network of neurons that is trained to provide the right output for some given inputs. The neurons have some weighted inputs and are responsible for simple operations, but the whole network can make parallel calculations due to its wide parallel structure. The neural network derives its computing power from its massive parallel distributed structure and from its ability to learn and, then to generalize. The generalization refers to the production by the network of reasonable outputs for inputs not encountered during training.

In our previous work [3], we proposed a multilayer feed forward perceptron model, trained with the supervised back propagation learning algorithm [31] to compute disparities. However, the results obtained are satisfactory in terms of accuracy, but they are not suitable for real time applications because the processing time needed for standard image sets is very high (see figure 15). In the present study, a multilayer feed forward model based on simple supervised learning procedure was adopted in order to improve the processing time. A detailed of this procedure can be found in section 2.1.3

The proposed neural network is composed of four layers (see figure 2). Each layer is responsible for different task (input, score computation, decision and

output). Since the neurons perform simple operations, their input weights and transfer functions are adjusted as follows:

The input layer associated to a windows W_1 (7x7=49 pixels) has 147 neurons of (49 neurons intensities, 49 neurons gradient magnitudes and 49 neurons orientations). This layer has the function to compute the scores DSI^d_I of intensities, DSI^d_G of gradient magnitudes and DSI^d_O of orientations for each one pixel of the window W_1 as given in equations 8, 9 and 10. The transfer function is $f(x)$ = abs(x) and the input weights are the same and equal to n_1. We obtain then for each value of the disparity d (d = 0..d_{max}) three (7x7) matrices of scores.

To compute the final DSI^d, the second layer adds the three correlation scores for each one pixel of the window (see equation 11). We obtain d_{max} + 1 matrices of 7x7 scores. The transfer function is linear and the input weights are identical and equal to n_2. In the third layer, for each value of d, all scores of the W_1 pixels are added and constitutes the score $SumDSI^d$ of the central pixel. Then, to the central pixel of the window is associated a vector of costs (Aggregation cost) AC = ($SumDSI^0$,$SumDSI^1$,…, $SumDSI^{dmax}$). The input weights are the same and equal to 1 and the transfer function is linear. In the fourth layer, the minimum cost amount of the d_{max} + 1 costs is chosen as the best score and defines the disparity d^* of the central pixel of the window. The input weights are the same and equal to 1 and the transfer function is linear $f(x)$= Min (x). Figure 3 illustrates how we compute the disparity d^*.

2.1.3. Learning Procedure

The neural correlation network must be trained with a supervised learning procedure before computing the minimum of $SumDSI^d$ values (best score) for each pixel. In general, neural supervised learning is based on the

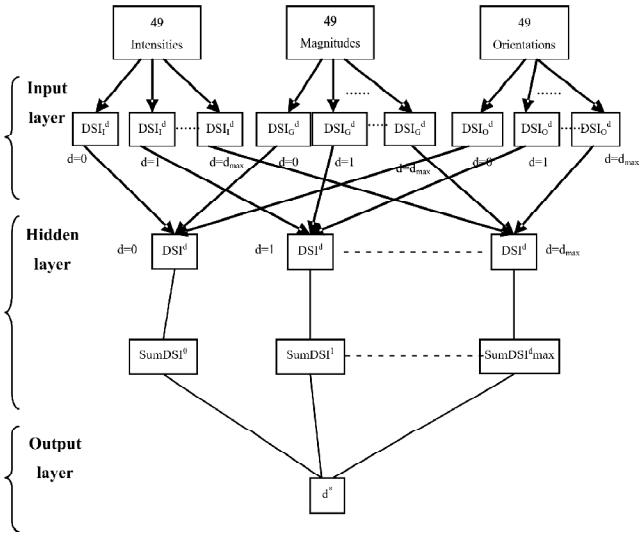

Figure 2: The Neural-DSI network architecture

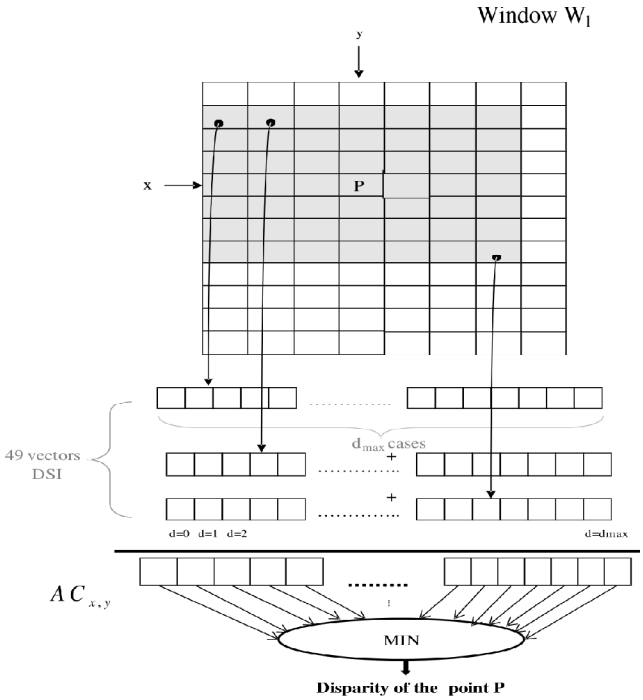

Figure 3: Disparity calculation d*

presentation to the network of a set of training examples having the structure [(features); (expected value)]. In our context, the feature designates a given training example of matching pixels (p_l, p_r). The expected value is the correlation score of (p_l, p_r). The first goal of learning procedure can be formulated as the search for the appropriate weights.

To prepare the training data, some points of interest are extracted from the stereoscopic pair of images and their attributes (gradients magnitude and orientations) are computed. These points are selected depending on their high values of intensity, gradient magnitude and orientation of the gradient vector. The matching of these points is done by using the normalized correlation *ZNCC* [2] considering the left image to the right image and vice versa. A valid match is considered only for those points that yield the best correlation score. These pixels are selected to train offline the network. During training, the differences of intensities, gradient magnitudes and orientations between two local windows (one for the left image, the other for the right) are fed to the network. Our method is based on exhaustive search of the best weights, in such manner to minimize the error of the matched pixels and to maximize the difference between the error of the matched and unmatched pixels. In our case, we defined two input weights: n_1 used in the first layer and n_2 used in the second layer. After the training, the network should have the ability to differentiate the matched pairs from unmatched ones.

In our experiment, we have trained our NN with number of epochs =100 which controls the number of

iterative presentation of examples. Experiments were conducted demonstrating that the increase of the number of epochs over the value 100 don't give better performances. Learning rate and momentum were set at 0.5 and 0.7 respectively.

2.2. Disparity Map Refinement

The resulting disparity map described above is not the optimal one because it still contains some noise and errors. We propose in the following our disparity map refinement method. Even if there exist more accurate techniques for the sub-pixel refinement in the literature [30], [34], they are computationally too expensive for real-time stereo vision.

2.2.1. Refinement Method

In the refinement method, we propose to use all scores obtained in the previous step. Instead of selecting in the window W_l only one disparity having the best score, we propose to select for each pixel n disparities corresponding to the best scores DSI^d. We assume that initial disparities of all pixels of the left image are computed. For each pixel p_l of the left image, we first verify if the disparity is dominant in the window W_l. If it is the case, this disparity will be considered as the final disparity and does not necessitate any refinement. Otherwise, we do a refinement which consists of selecting in W_l window three best scores for each pixel (see figure 4). Three best disparities are then associated to the pixel p_l instead of one disparity. The proposed process consists of applying a vote in order to choose the dominant disparity in the associated W_l using the three disparities of the central pixel p_l and its 48 neighboring pixels.

Figure 4 illustrates how we compute the new disparity using the three best disparities for each pixel when we apply the vote process. The disparity that will obtain the highest number of points will be considered as the new disparity.

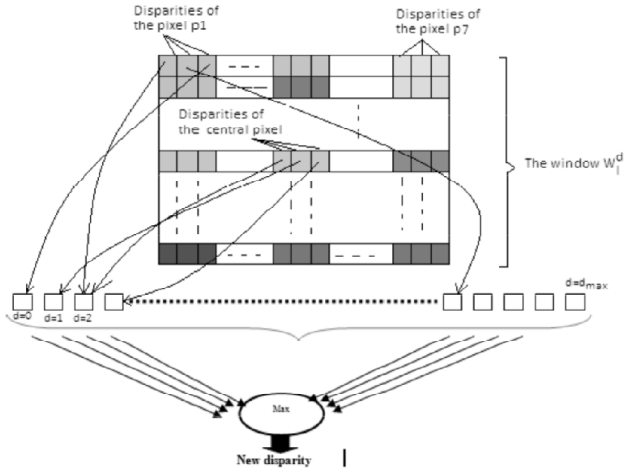

Figure 4: Disparity computing with our refinement method

For the definition of the parameter **n,** we decided to use an experimental evaluation of over existing stereo datasets [32]. The number **n** = 3 corresponding to the three best disparities values has been selected. If we take **n** greater than 3 it did not lead to an increase in accuracy but an increase in processing time (figure 5). For **n** = 4 the accuracy increases just 0,029% while the processing time increases of 0,07s. This is the main reason why increasing the n value beyond a certain value does not improve accuracy any further.

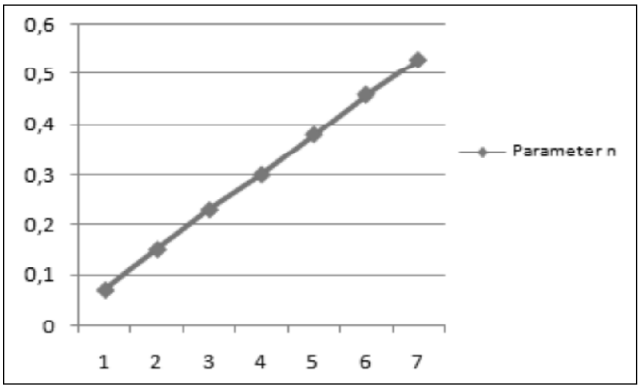

Figure 5: Processing time (second) obtained by the refinement method with different values of n

2.2.2. Processing of Region Boundaries Problem

The region boundaries problem occurs when the pixels of the same window belong to two different regions (different objects see figure 6). Indeed, the pixels of the two sides of the contour usually have different disparities. Consequently, only the pixels belonging to the same region (object) must be taken into account in the computation of the new disparity (final disparity).

To solve this problem, instead of using all pixels of the window W_l, in the vote process described above, we add a criterion which eliminates from the vote process all pixels of the window W_l located in the second region detected using the gradient magnitude informations.

With this improvement, the disparity map is more accurate because only pixels of the window belonging to the same region (object) are involved in the calculation of the new disparity. Figure 6 illustrates how we compute

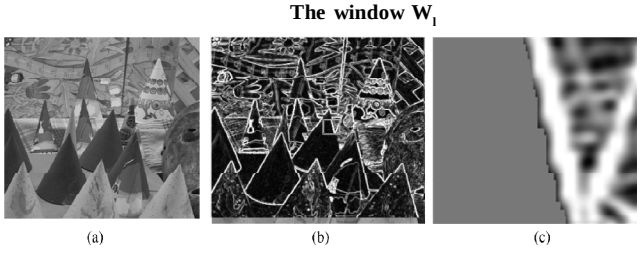

Figure 6: Example of region boundaries problem: (a): The reference image, (b): Case where the window W_l contains pixels of two regions, (c): The blue color represents the pixels of the same region of the window W_l involved in the computation of the new disparity

the new disparity in the case where the window W_l is positioned on two regions (objects).

2.2.3. Disparity Map Smoothing

Finally, in order to keep the good trade-off between accuracy and processing time, a simple median filter is applied for smoothing the disparity map. The median filter is a robust method, often used to remove the impulsive noise known for its salt and pepper noise from an image. The median is calculated by first sorting all the pixel values from the surrounding neighbourhood into numerical order and then replacing the pixel being considered with the middle pixel.

3. FPGA IMPLEMENTATION

In order to reduce the computing time, we propose to implement algorithms of disparity map computation on FPGA: Field Programmable Gate Array. Indeed the FPGAs capability has attracted researchers in computer vision. The use of FPGAs is now the most suitable and practical choice for hardware implementation. They are cheap and perform extremely well [26].

3.1. Related Work

Various examples of stereo vision algorithms implemented on FPGAs have been reported in the literature. Yi [43] proposed a stereo vision system based on a Xilinx Virtex II, which uses the SAD algorithm. The system can process images with a size of 270x270 at a frame rate of 30 frames per second. But the maximum disparity is 34 pixels. In Miyajima [24], an FPGA-based system achieves 19 frames per second for 640x480 image pair resolution with 80 levels of disparity. Han[12] have proposed a system that can process images with a resolution of 640x480 at a frame rate of 60 frames per second and a maximum disparity of 128 pixels for the use in a household mobile robot. In Khaleghi [15] an FPGA-based module reaches 10 frames per second for 640x480 pixels resolution image with a disparity range of 30 levels. He implemented the SAD method using three levels of recursion developed by [9]. In Woodill [40], a 3D-vision system implemented for tracking people, reaches 30 frames per second for 512x480 pixel resolution image pair with 52 levels of disparity. The system described by Murphy [23] proposes an implementation of the census transform algorithm on Spartan-3 FPGA. It can process 320x240 images at 150 frames per second, but only with a disparity range of 20 pixels. In Ambrosch [1], a performance of 600 frames per second is reported for 450 x375 pixel resolution image pair with a disparity range of 100 pixels. He implemented the SAD algorithm using Altera Quartus II family. The disparity range is split into n partitions and the SAD matching costs are calculated for each partition. The system proposed by Georgoulas [9] achieves 768 frames per second for 640 x480 pixel resolution image pair with a disparity range of 80 pixels. He implemented the color SAD window based algorithm on the Stratix IV family.

Other works use a graphics processing unit (GPU) for the stereo matching. Yang [42] proposed a system using the GPU achieving 50-70 M disparity evaluations per second, using a multiresolutions approach. When using a resolution of 512x512 for the input images and the resulting disparity map, the system reaches a frame rate of 4.8 frames per second at a disparity range of 100 pixels. Another GPU-based stereo vision system was proposed by Prehn [29]. He used a GeForce 8800 GTS graphics card reaching 18 frames per second at an image size of 450x375 when using a block size of 7x7 for the SAD algorithm and a disparity range of 59 pixels. (Gong, 2007) proposed a system using GPU achieving 178 ms at disparity range of 16 pixels for Tsukuba pair. Another GPU-based system was proposed by Boufarguine [7]. He used CUDA API reaching 27 frames per second at a Tsukuba image using belief propagation algorithm and disparity rang of 16 pixels.

Finally, it must be noted that most methods reported previously use the simple algorithm SAD by proposing different architectures which generally result to semi-dense disparity estimation. They are designed to perform better in real-time speeds term and to present a poor quality in disparity map.

3.2. Proposed Architecture

The architecture corresponding to our method is split into three major pipeline stages: pre-processing, calculation and post-processing stages. The interfaces between them are formed by internal memory.

The first stage is the input stage which supplies the image data for the computation. The system uses two on chip buffers, which hold the part of the image which is being searched (one buffer for the left image and one for the right image). The size of the buffers depends on the correlation window size and the disparity range. In our case the window size used is 7x7 and the disparity range is 50. At this stage, the Sobel operator is used to compute the image gradient. It is applied in parallel to both images in x and y direction.

The calculation stage concerns the Neural-DSI algorithm which computes the initial disparity map. The calculation stage decomposes the disparity range into r separate partitions where each partition contains $d_r = d_{\max} / r$ disparity levels. This way, the calculation stage can compute each partition in a separate calculation round. The time required for the computations of each image lines is now defined by the number of separately calculated partitions r and the logic consumption can be scaled by the number of pixels d_r in each partition.

Finally, the initial disparity map is refined in the post-processing stage. For this, we propose architecture for the median filter. All three steps are fully implemented in hardware based on FPGA devices of Virtex-II family using hardware description language VHDL. For this, we have proposed architectures for:

- The Sobel operator: used to calculate the image gradient.
- Neural-DSI: used for the computing the initial disparity map.
- The Median filter: used for the refinement of the initial disparity map.

3.2.1. Sobel Operator Architecture

In our work, we used Sobel operator for the gradient magnitude computation which is given by the following equation:

$$|G(i,j)| = |G_x(i,j)| + |G_y(i,j)| \quad (12)$$

In order to compute G_x and G_y for each pixel in the left and right images, we applied the 3x3 mask for each pixel such as:

$$G_x = \begin{bmatrix} I(i-1,j-1) + 2*I(i-1,j) \\ I(i-1,j+1) - I(i+1,j-1) \\ -2*I(i+1,j) - I(i+1,j+1) \end{bmatrix} \quad (13)$$

$$G_y = \begin{bmatrix} I(i-1,j-1) + 2*I(i,j-1) \\ +I(i+1,j-1) - I(i-1,j+1) \\ -2*I(i,j+1) - I(i+1,j+1) \end{bmatrix} \quad (14)$$

The general architecture of the Sobel operator is illustrated by the figure 7. The circuit input is 8 bits per pixel from a bus of eight bytes and 8 bits for the threshold value. The output circuit is an 8 bits value equal to 255 if the pixel belongs to an outline and 0 otherwise.

The Sobel operator involves convolution of the input image over two convolution masks; the masks hold data values between -2 and 2; thus the overall convolution does not need to involve a multiplier. A set of additions and subtractions, depending on the mask value (i.e. one addition for a value of +1, two additions for a value of +2, and so on) was used. By avoiding the costly multiplication operation, a higher frequency as well as fewer clock cycles could be obtained, at the cost of quality however.

The various units of this architecture are shown in figure 7.

Figure 8 shows the detailed architecture of the processing unit

The above circuitry is a highly parallel structure requiring 6 clock cycles, 9 adders, 4 shifters, 2 substractors, 1 comparator and 1 decoder. The running time for one iteration is 18,204 ns.

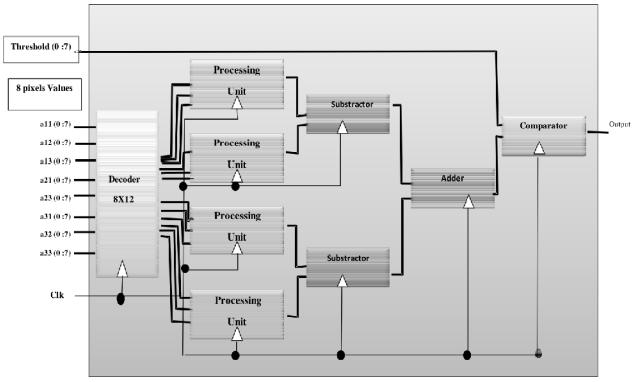

Figure 7: Proposed architecture of Sobel operator

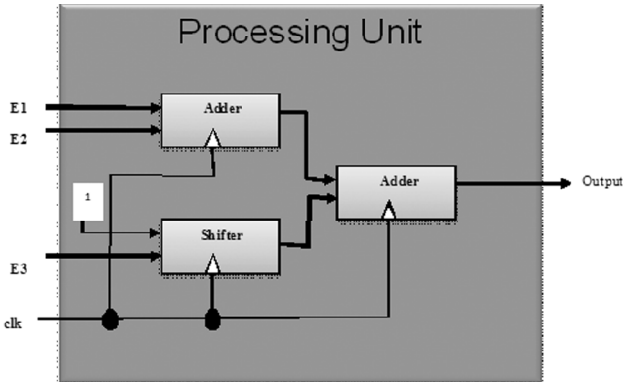

Figure 8: Processing unit architecture

The proposed architecture has been implemented using the FPGA circuits (Xilinix Virtex-II XC2V40). It has been then simulated to prove its functionality. The analytical specifications of the target device are shown in table 1.

Table 1
Device Utilization Summary (Estimated Values)

	Used	*Available*	*Utilization*
Number of Slices	140	256	54%
Number of Slice Flip Flops	136	512	26%
Number of 4 input LUTs	136	512	26%
Number of bonded IOs	81	88	92%
Number of GCLKs	4	16	25%

3.2.2. Neural-DSI Architecture

In order to optimize the processing time of the disparity map calculation method, we have implemented the most time consuming part of our architecture on Xilinix FPGA circuits (Virtex-II XC2V80).

For each pixel $p_l(x_l, y_l)$ in the left image, the disparity computation will concern all pixels of a window W_l centered on p_l (section 2.1.2) . In our case, we used a

shiftable window of 7 x7 size (Line x column). To increase the processing speed of the neural-DSI method and more parallelism, when we compute the disparity of $p\,(i, j+1)$, we perform the operations shown in Fig. 9 (a). Thus, we compute the disparity for one column at a time rather than the entire window. The same principle is used to calculate the disparity of the pixel $p(i+1,j)$. We do the computation for only one line at a time rather than the entire window (see figure 9(b)). By using these operations, the redundancy in computation is completely removed.

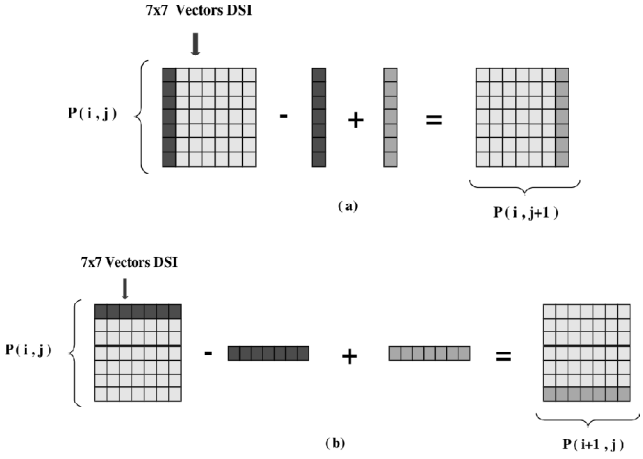

Figure 9: Illustration of computing operations used in Neural-*DSI* method

In this work, we address the computing of the disparity map problem by using the highly parallel structure shown in figure 10. It is composed of three parts:

- Memory Unit (storage): composed of two blocks of RAM 128x40 and one block RAM 16x80. Where each block RAM 128x40 is composed of 5 RAM 128x8 and the block RAM 16x80 is composed of 5 RAM 16x16.
- Computing Unit: composed of 5 adders and 5 subtractors.
- Control Logic (state machine): composed of 4 comparators.

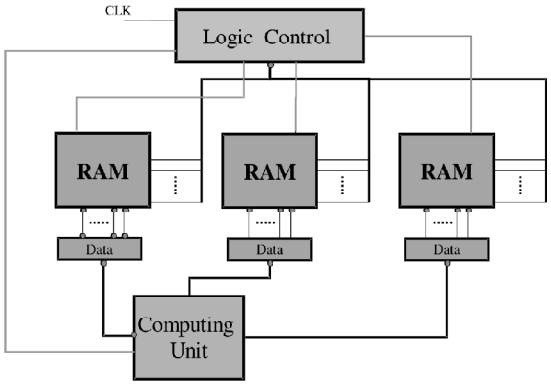

Figure 10: Proposed architecture of Neural-DSI method

Table 2 shows the percentage of consumed resources of the FPGA device by the modules of the proposed architecture of Neural-DSI method.

Table 2
Device Utilization Summary (Estimated Values)

	Used	*Available*	*Utilization*
Number of Slices	410	512	80%
Number of Flip Flops	219	1024	21%
Number of 4 input LUTs	816	1024	79%
Number of bonded IOs	81	92	88%
Number of GCLKs	4	16	25

The running time of the disparity calculation corresponding to one iteration is about 91,05 ns. The proposed architecture has been implemented using the FPGA circuits (Xilinix Virtex-II XC2V80) and then it has been simulated to prove its functionality using the ISE simulator of Xilinx Virtex-II Family. The figure 11 shows an example of the hardware simulation result of the Neural–DSI architecture.

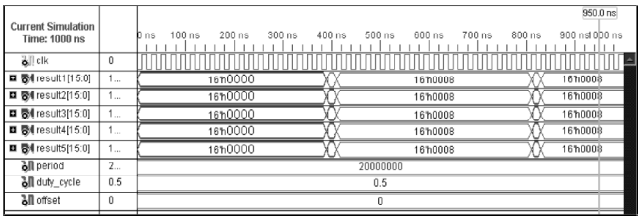

Figure 11: Hardware simulation of the Neural-*DSI* architecture

3.2.3. Median filter

Median filter is a robust method to remove the impulsive noise from an image. It is a compute intensive operation, so it is hard to implement it in real time. Different approaches have been proposed in the literature for the median filter implementation with FPGAs [22, 25, 35]. In our case we implemented the one proposed by Vega-Rodriguez [25] since it needs a very lower number of basic nodes (19 instead of 27).

In our case we used 3x3 mask of the median filter. The circuit input is 8 bits per pixel from a bus of 9 bytes. The circuit output is an 8 bits value equal to the median value. Figure 12 shows the filter median architecture sorting 9 pixels and selecting the median.

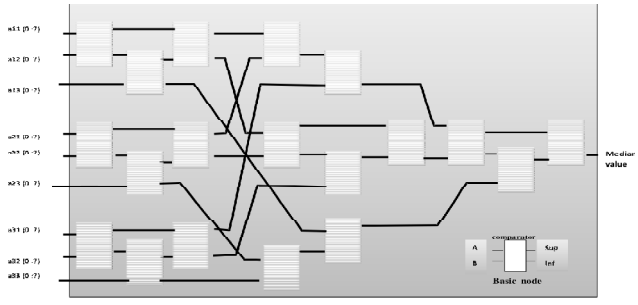

Figure 12: Proposed architecture of Median Filter

Figure 13 shows the detailed architecture of the basic node of the median filter.

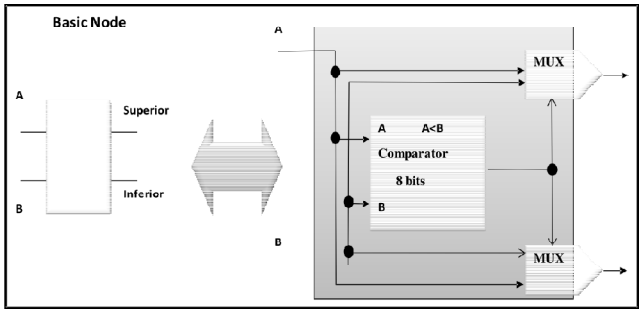

Figure 13: Basic node architecture

Table 3 shows the percentage of consumed resources of the FPGA device by the modules of the proposed architecture of the median filter using FPGA circuits (Xilinix Virtex-II XC2V80).

Table 3
Device Utilization Summary (Estimated Values)

	Used	*Available*	*Utilization*
Number of Slices	220	512	42%
Number of Flip Flops	236	1024	23%
Number of 4 input LUTs	408	1024	39%
Number of bonded IOs	81	92	88%
Number of GCLKs	6	16	37%

So, the running time for one iteration is 21.285 ns.

4. EXPERIMENTS RESULTS

In this section, we describe the experiments conducted to evaluate the performance of the proposed method. Our aim is to obtain an accurate disparity map and a fast runtime which is the requirement of any obstacle detection system of autonomous mobile robot navigation. To this purpose, an extensive performance evaluation and comparison between different methods is proposed. The two criteria used for the evaluation are then accuracy and computation cost.

Several parameters have been mentioned and discussed in the next. Qualitative tests through disparity map observation were carried out with four stereo couples [32] to find the influence and appropriate values of those parameters.

Figure 14 illustrates the results of the disparity map obtained. From left to right: the first column images are the reference images, the second column images are the Ground truths, the third column images are the initial disparity maps obtained and the four column are the final disparity maps obtained after the refinement step.

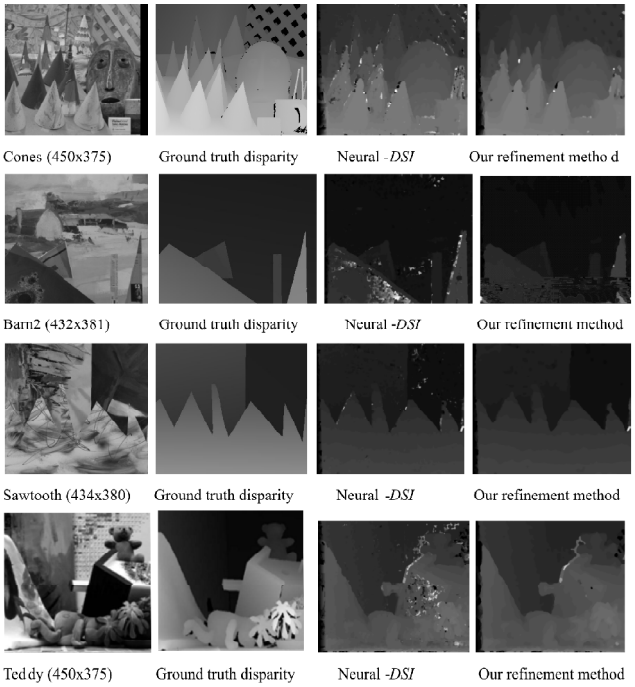

Figure 14: Disparity map obtained with our method, From left to right: Images reference, Ground truths, initial disparity maps, and final disparity maps

In order to compare the results of different algorithms, we adopt a method similar to that of Scharstein and Szeliski, [33]. Parameters ALL and NOCC are defined according to the Middlebury website. ALL is the error computed on the whole image and NOCC is the error computed on the whole image excluding the occluded region. Among the quality measures proposed by Scharstein and Szeliski [33] in their paper we adopted the percentage of bad matching pixels between the computed disparity map $d_C(x, y)$ and the ground truth map $d_T(x, y)$:

$$PBP = (1/N \Sigma(|d_C(x, y) - d_T(x, y)| > \delta_d)) \qquad (15)$$

Where δ_d is the error disparity deviating from the ground truth more than 1 pixel. Table 4 shows the results in term of accuracy obtained by our method on some images available in the Middlebury website.

Table 4
Accuracy According to the Methodology Defined by the Middlebury Web site

Cone		*Teddy*		*Tsukuba*		*Venus*		*Aver-age bad Percent*	*Accur-acy*
ALL	*NOCC*	*ALL*	*NOCC*	*ALL*	*NOCC*	*ALL*	*NOCC*		
15.2	7.97	21.3	14.3	5.69	3.78	3.84	2.6	9.3	90.07

In order to study the efficiency of the combination of the neural network and DSI concept, two other approaches were implemented: The neuronal method [3]

called (Neural) and the DSI method described in [5] called (DSI). We applied these methods on four images (Cones, Barn2, Sawtooth, Teddy) of standard data sets available on the Middlebury stereo evaluation website [32].

Similarly to the evaluation of accuracy, figure 15 illustrates the results of the processing time (second) obtained for three selected methods. The three selected methods were implemented using the C++ language and the timing tests were performed on a Personal computer PC, 2.5 GHZ. We can clearly see (figure 15) that our approach (Neural-DSI) is relatively the fastest among DSI and neural methods.

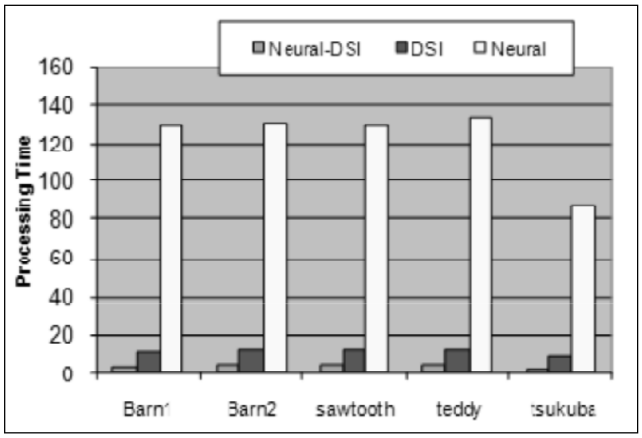

Figure 15: Processing Time (seconds) of Neural-DSI, Neural and DSI methods on the five image pairs

We studied also the influence of window size on the accuracy of the proposed method. Figure 16 shows the disparity map obtained after applying our refinement method for the Barn1 image pair for different sizes of the window. Greater the size of the window W, better the computed map disparity is. Nerveless, the computation time is also very high when the size of W is large. Figure 17 illustrates the variation of time processing for three methods Neural, DSI and our method (Neural-DSI).

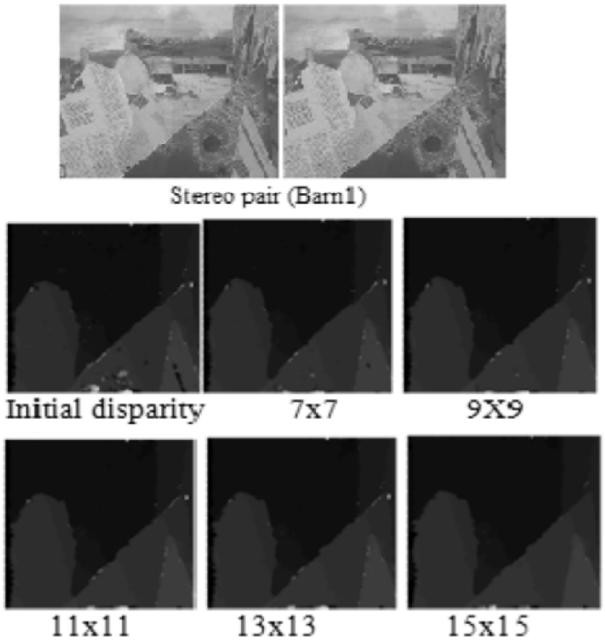

Figure 16: Application of the refinement method on Barn1 image for different window sizes

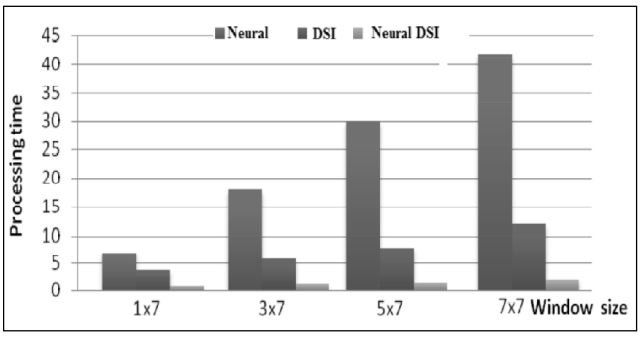

Figure 17: Processing time (seconds) of the Neural, DSI and Neural-DSI methods for different window sizes

Experiments are conducted in order to study the influence of d_{max} values on the performance of the Neural-DSI method. We used different values of this parameter for map disparity estimation of four stereo images pairs with 1x7 window. Figure 18 illustrates the processing times obtained for the Map image and show that the Neural-*DSI* method is the fastest. Indeed, the processing time obtained is less than 0, 2 seconds.

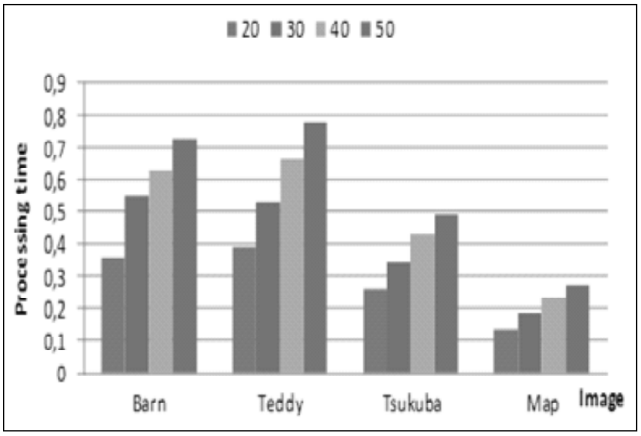

Figure 18: Processing time(s) of disparity map computing of Neural- DSI with different values of d_{max} on four images pairs with 1x7 window

To demonstrate the importance of the use of FPGA circuits, table 5 illustrates the processing time obtained for each component in traditional implementation (Soft) with 7 x7 window where:

- Pair1, Pair2, pair3 are respectively Barn1 (432x381), Teddy (450x375), Tsukuba (384x288).
- Methods 1, 2, 3 correspond respectively to Gradient (Sobel), Neural- DSI, and Median Filter.

Table 5
Processing time (ms) for Software Implementation

Method	*1*	*2*	*3*
Pair1	147	730	468
Pair2	163	780	470
Pair3	107	490	312.5

Table 6 illustrates the processing time obtained for each component using FPGA implementation. Not surprisingly the running times obtained with the use the FPGA are better. The processing time for a Tsukuba image pair was 490ms. It seems obvious that even with the algorithmic and software optimizations, the processor-based system cannot outperform the FPGA-based solution. All the reasons make FPGA implementation preferable.

Table 6
Processing Time (ms) for FPGA Implementation

Method	*1*	*2*	*3*
Pair1	2.99	14.99	3.46
Pair2	3.07	15.36	3.55
Pair3	2.01	10.07	2.32

4.1. Discussion and Comparison

This section presents a comparison between the proposed method (Neural-DSI) and other state-of-art methods. The results obtained by our algorithm are better than some methods reported in the literature [1][26]. Table 7 shows a comparison of stereo vision implementation reported in the literature in terms of computation time. The description of the systems introduced here is restricted to the basic matching strategy, the image size, the disparity range search (d_{max}) and the processing time achieved. All performance data are taken from the authors' papers.

Compared to other FPGA-based approaches, the obtained results are better than some methods reported in literature as shown in table 7. We can also see that the results of Ambrosch [1], Georgoulas [9] are faster but their disparity maps are less accurate than our disparity map as shown in figure 19.

Since the used stereo vision algorithms are very different and don't used the quality metrics described in the Middlebury website, we could not make direct comparisons quality images obtained by each method. Except for [1] who obtained an accuracy of 62.36%. In our case, with our algorithm for the same image and the same window size we obtained 90.6%.

Table 7
Comparison of Stereo Vision Implementations

Author	*Time (ms)*	*Algorithm*	*Image Size*	d_{max}
Miyajima *et al*. [24]	52,63 ms	SAD	640 x 480	80
Niitsuma *et al*. [27]	33,34 ms	SAD	640 x 480	27
Han *et al*. [12]	16,67 ms	SAD	640 x 480	128
Yi *et al*. [43]	33,34 ms	SAD	270 x 270	34
Woodill *et al*. [40]	33,34 ms	SAD	512 x 480	52
Georgoulas *et al*. [9]	1,3 ms	SAD	640 x 480	80
Ambrosch *et al*., [1]	1,67 ms	SAD	450 x 375	150
Khaleghi, *et al*. [15]	100 ms	SAD	640 x 480	30
Yang *et all*. [41]	62,5 ms	Belief Propag.	320 x240	16
Proposed impl.	10 ms	Neural-DSI	384 x 288	50
Murphy *et al*. [23]	6.67 ms	Census. Transf.	320 x240	20
Masrani *et all* [20]	33,34 ms	LW Phase Corr.	640 x 480	Dyn.

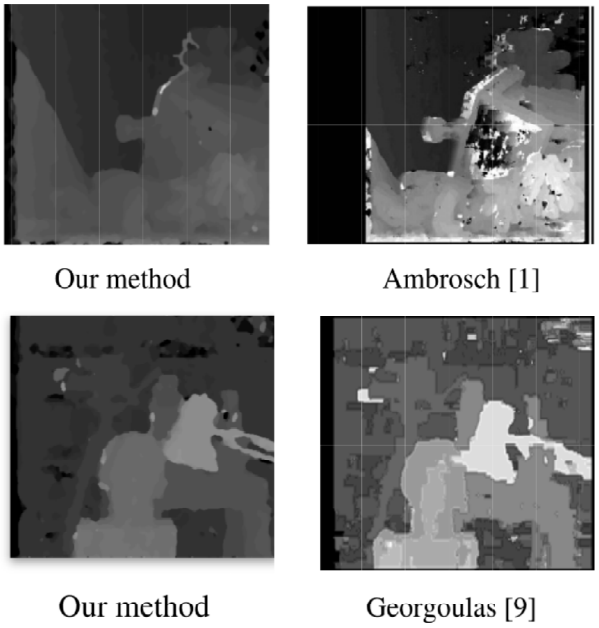

Figure 19: Disparity maps obtained by our method, Ambrosch [1] and Georgoulas [9] methods

5. CONCLUSION

The stereo correspondence problem remains an active area for research. In this paper, we proposed a hardware implementation for a new real-time stereo vision method using FPGA for the calculation of disparity map based on the combination of the neural network and *DSI* data structure. As seen in the experiments results, we proposed architectures based on fully parallel-pipelined blocks in order to achieve maximum processing time with accurate disparity map. Overall, the proposed approach showed the capabilities to improve the accuracy and can be regarded as an interesting trade-off between accuracy and speed. The three architectures of our method have been tested. Thus, in the future work, and in order to achieve

maximum processing time we will implement the three parts of our method on single FPGA circuit XC2V1000 which is more powerful.

REFERENCES

[1] Ambrosch, K., Humenberger, M.,Kubinger, W., Steininger,. A., SAD –based stereo matching using FPGAs, in Embedded computer vision part II, Springer, pp. 121-138, 2009.

[2] Baha, N., Larabi, S.,. Obstacle Detection from Uncalibrated Cameras. *In Proc of PCI2008*, Greece, 152-157, 2008.

[3] Baha, N., Larabi S., Disparity Map Estimation with Neural Network, in IEEE ICMWI'2010, *International Conference on Machine and Web Intelligence*. 3-5 October 2010, pp. 282-285.

[4] Banno, A., Ikeuchi, K., Disparity map refinement and 3D Surface smoothing via directed anisotropic diffusion. 12 th ICCV Workshops, 1870-1877, 2009.

[5] Binaghi, E., Gallo, I., Fornasier, C., Raspanti, M., Growing aggregation algorithm for dense two-frame stereo correspondence. *In 1st Int. Conf. on Computer Vision Theory and Application*, 326-332, 2006.

[6] Bobik,A., Intille,S., Large occlusion stereo. Intern. *Journal on Computer Vision* **33**, 181-200, 1999.

[7] Boufarguine M., Baklouti, M., Guitteny, V.,Couvert, S., Real time Disparity Estimation Using CUDA'S API. VISAPP 2009, 2009.

[8] Di Stefano, L., Marchionni, M., Mattoccia, S., A fast area-based stereo matching algorithm. *Image and Vision Computing*, **22**(12), 983-1005, 2004.

[9] Georgoulas, C., Andreadis, I., A real-time occlusion aware hardware structure for disparity map computation, ICIAP2009, LNCS, Springer_Verlag, pp. 721-739, 2009.

[10] Gong, M., Yang, Y.H., Real-time stereo matching using orthogonal reliability-based dynamic programming. *IEEE Transaction on Image Processing* **16**(3), 879-884, 2007.

[11] Gonzalez, R.C. and Woods, R.E., Digital image processing. Prentice Hall, Upper Saddle River, New Jersey, 2002.

[12] Han, D. and Hwang, D., A novel stereo matching method for wide disparity range detection, *Lect. Notes Comput. Sci.* 3656, 643–650, 2005.

[13] Han, D., Real-time object segmentation of the disparity map using projection based region merging., Scene reconstruction, Pose estimation and tracking , Book edited by R. Stolkin, ISBN 978, 530, 2007.

[14] Kanade, T., Okutomi, M., A stereo matching algorithm with an adaptive window: theory and experiment. *IEEE Trans. Pattern Anal. Machine Intell.*, **16**, 920-932, 1994.

[15] Khaleghi, B., Ahuja, S.,Wu, J. Q. M., A New Miniaturized Embedded Stereo-Vision System (MESVS_I), In proceedings of Canadian Conference on Computer and Robot Vision, pp. 26-33, 2008.

[16] Kim J., Lee K., Coi B. and Lee S., A dense stereo matching using two-pass dynamic programming with generalized ground control points, *In Proc. Conf. Computer Vision and Pattern Recognition*, 2005, pp. 1075-1082.

[17] Kolmogorov, V., Zabih, R., What energy functions can be minimized via graph cuts?. ECCV(3), *Lecture Notes in Computer Science*, pp. 65-81, 2002.

[18] Kumar, S., Chatterji, B., Stereo matching algorithms based on fuzzy approach. *Int. Journal in Pattern Recognit. Artif. Intell.* **16**, 7, 883–899, 2002.

[19] Maas, R., Haar Romeny, B., Viergever, M. Area-based computation of stereo disparity with model_based window size selection. *Computer Vision and Pattern Recognition* (CVPR), 106-112, 1999.

[20] Masrani, D. K., MacLean , W. J., A Real-Time Large Disparity Range Stereo-System using FPGAs, *Proc. IEEE Intern. Conf. Comput. Vis. Syst.*, 2006.

[21] Mattocia, S. A. locally global approach to stereo correspondence. ICCV, 12 th Intern. Conf. on Computer Vision Workshops, 2009, 1763-1770.

[22] Morcego, B., Frau, J. Català, A. Suavizado de Imágenes en Tiempo Realmediante Filtrado por Mediana Utilizando Arrays Sistólicos, Proc. of VII DCIS, pp. 545-546, 1992.

[23] Murphy, C., Lindquist, D., Rynning, A. M., Cecil, T., Leavitt, S., Chang, M., 2007. Low Cost Stereo Vision on an FPGA, Proc. 15th IEEE Symp. FPGAs Cust. Comput. Mach.

[24] Miyajima, Y., Maruyama, T., A Real-Time Stereo Vision System with FPGA. Lect. Notes Computer Science 2778, 448-457, 2003.

[25] Vega-Rodríguez, M. A., Sánchez-Pérez, J. M. , Gómez-Pulido, J. A. An FPGA-Based Implementation for Median Filter Meeting the Real-Time Requirements of Automated Visual Inspection Systems, 2002**.**

[26] Nalpantidis, L., Sirakoulis, G., and Gasteratos, A., Review of stereo matching algorithms: from software to hardware. *International Journal of Optomechtonics* **2**: 435-462, 2008.

[27] Niitsuma, H. and Maruyama, T., Real-time detection of moving objects, *Lect. Notes Comput.Sci.* 3203, 1155–1157, 2004.

[28] Ogale, A., Aloimonos, Y., Shape and the stereo Correspondence Problem. *Inter. Jour. on Computer Vision, IJCV* **65**, 3, 147-1758, 2005.

[29] Prehn, S., GPU Stereo Vision, Project Thesis, Robotics Research Lab, University of Kaiserslautern, 2007.

[30] Psarakis, E. Z., Evangelidis, G. D. A generic implementation framework for FPGA based stereo matching, in Proceedings of the IEEE Region 10th Annual Conference on Speech and Image Technologies for Computing and Telecommunications, 1997.

[31] Rumelhart, H., Hinton, G. E., Williams, R. J., Learning internal representation by error propagation. *Parallel Distribut. Process.* 318–362, 1986.

[32] *http://www.middlebury-edu/stereo.*

[33] Scharstein, D., Szeliski, R., A taxonomy and evaluation of dense two-frame stereo correspondence algorithms. *Int. Journal on Computer Vision*, **47**, 7–42, 2002.

[34] Shimizu, M., Okutomi, M., Precise sub-pixel estimation on area based matching, in: Proceedings of the eight IEEE International Conference on Computer Vision, 2003.

[35] Smith, J. L, Implementing Median Filters in XC4000E FPGAs. *Xcell*, **23**(4) 1,6, 1996.

[36] Tappen M.,Freeman W., Comparison of Graph cuts with Belief Propagation for Stereo, using identical MRF, *Parameters. Inter. Conf. on Computer Vision*, ICCV 2003.

[37] Twardowski, T., Cyganek, B., Borgosz, J. Gradient based dense stereo matching, *Lecture Notes in Computer Science* **3211**(8), (2004), 721-728.

[38] Vanetti, M.,Gallo,I.,Binaghi,E.,. Dense Two-Frame Stereo Corresponence by Self-Organizing Neural Network. in Image analysis and processing, LNCS, Springer, 2009, 1035-1042.

[39] Veksler, O., Extracting dense features for visual correspondence with graph cuts. *Proc. of the IEEE Computer Society Conference on Computer Vision and pattern Recognition*. Pp 689-694, 2003.

[40] Woodill, J. I., Buck, R., Jurasek, D., Gordon, G., Brown, T., 3D Vision: Developing an Embedded Stereo-Vision Sustem. *IEEE Computer* **40**(5), 106-108, 2007.

[41] Yang, Q., Wang, L., Yang, R., wang, S., Liao, M. Nister D., Real-time Global Stereo Matching using Hierarchical Belief propagation. *The British Machine Vision Conference (BMVC)*, 2006.

[42] Yang, R., Pollefeys, M., Multi-resolution real-time stereo on commodity graphics hardware, proc. Conf. Comput. Vis. Pattern Recognt, 2003.

[43] Yi, J., Kim, J., Li, L., Morris, J., Lee, G, Leclercq, P., Real-Time three dimensional vision, Lect. *Notes Comput. Sci*. 3189, 309-320, 2004.