# An FPGA stereo matching unit based on fuzzy logic

M. Pérez-Patricio, Abiel Aguilar-González, M. Arias-Estrada, H. R. Hernández-de León, J. L. Camas-Anzueto, J.A. de Jesús Osuna-Coutiño

# An FPGA stereo matching unit based on fuzzy logic

M. Pérez-Patricio[a], Abiel Aguilar-González*[b], M. Arias-Estrada[b], H. R. Hernández-De León[a], J. L. Camas-Anzueto[a], J. A. de J. Osuna-Coutiño[a]

[a] *Instituto Tecnológico de Túxtla Gutiérrez, Tuxtla Gutiérrez, México*
[b] *Instituto Nacional de Astrofísica, Óptica y Electrónica, Cholula, México*

**Abstract**

The stereo matching is one of the most widely used algorithms in real-time image processing applications such as positioning systems for mobile robots, three-dimensional building mapping and both recognition, detection and three-dimensional reconstruction of objects. In order to improve the runtime, stereo matching algorithms often have been implemented in dedicated hardware such as FPGA or GPU devices. In this article an FPGA stereo matching unit based on fuzzy logic is described. The proposed method consists of three stages: first, three similarity parameters inherent to each pixel contained in the input stereo pair are determined; later, these parameters are submitted to a fuzzy inference system that determines a value of fuzzy-similarity; finally, the disparity value is determined as the index for the maximum value of the fuzzy-similarity values (zero up to $d_{\max}$). Dense disparity maps are computed at a rate of 76 frames per second for input stereo pairs of 1280x1024 pixel resolution and a maximum expected disparity equal to 15. The developed FPGA architecture provides reduction of the hardware resource demand; up to 67,384, minimum 9,788 for logic units, up to 35,475, minimum 11,766 for bits of memory. Increases the processing speed; up to 78,725,120, minimum 14,417,920 pixels per second and outperforms the accuracy level of most of real-time stereo matching algorithms reported in the literature.

*Keywords:* Stereo matching, Fuzzy logic, FPGA

## 1. Introduction

Depht values of the points contained in an image is one of the most used tasks of the computer vision systems and has been used in several applications such as positioning systems for mobile robots and both recognition, detection and three-dimensional reconstruction of objects [1–7]. In stereo vision technique the correspondence between stereo pairs and the geometrical configuration of the stereo camera allows to obtain images of depth called disparity maps. In order to determine a disparity map it is necessary to measure the similarity of the points contained in the stereo pair. Techniques to determine these similarities are divided in two categories: area-based algorithms [8, 9] and feature-based algorithms [10, 11].

Area-based algorithm use the color value of the surrounding pixels to the interest pixel and produce dense disparity maps, i. e., these compute the disparity value for each pixel in the input stereo pair. The main characteristic of area-based algorithms is mathematical simplicity and low runtime. On the other hand, feature-based algorithms are based on specific interest points and are more stable against changes of contrast, enviroment conditions and illumination due to these represent the geometric properties of the input stereo pair. However, these algorithms possess high runtime and high mathematical complexity.

## 2. Related works

FPGA devices allow the implementation of dedicated computational architectures that can accelerate algorithms. In previous work there has been several FPGA stereo matching architectures reported in the literature [12, 13]. However, in most of these FPGA architectures the authors have been concentrated in the accuracy level and not in the performance regarding to hardware resource consumption, which is an important parameter in mobile autonomous applications in which the use of small FPGA devices and small power sources are required.

### 2.1. FPGA implementations

The system presented in [14] consists in a 4×4 array of FPGAs connected in mesh type configuration, authors use a maximum total of near 35,000 LUT of 4 inputs, allowing to process 40 frames per second for images of 320×240 pixel resolution. In [15], a structure based on four FPGAs Virtex 2000E of Xilinx is presented, obtaining dense disparity maps at a speed of 40 frames per second for images of 256×360 pixel resolution. In [16], the use of a single FPGA is proposed, the developed system processes images at 30 frames per second using images of 640×480 pixel resolution. The architecture developed in [17], uses a technique based on SAD to calculate the optical flow efficiently, the system generates dense disparity maps at speeds superiors to 800 frames per second for images of 320×240 pixels using a correlation window of 7×7 and a maximum expected disparity equal to 121.

*Corresponding author
*Email address:* abiel.aguilar@ittuxtlagutierrez.edu.mx (Abiel Aguilar-González*)

A modification of SAD is shown in [18], the authors of this work synthesize diverse versions of SAD to determine the needs and the performance of the hardaware resources, by decomposing the correlation window of SAD in rows and columns using buffers a saving of resource of around 50% is reached. Using different forms of windows, the high consumptions of memory decreases without any detriment of the quality. Disparity maps are calculated at speed of 122 frames per second for images of 320×240 pixels and a maximum expected disparity equal to 64.

The architecture in [19] uses four FPGAs to conduct a rectification in real-time, later, a verification of left-right consistency was applied in order to improve the quality of the produced disparity map. Speeds of 30 frames per second are reached for images of 640×480 pixel resolution and a maximum expected disparity equal to 128. In [9] an FPGA correlation-edge distance approach is proposed. Speeds of 76 frames per second are reached for images of 1280×1024 pixel resolution and a maximum expected disparity equal to 15. By using a geometric feature, the euclidean distance between the selected point and the nearest left edge, the developed FPGA architecture provides a improvement over others conventional correlation-based stereo matching algorithms.

### 2.2. *Fuzzy logic approaches*

In [20], similarity measure for stereo matching based on fuzzy relations is proposed. The strength of relationship of fuzzy data of two windows in the left image and the right image is determined by applying fuzzy aggregation operators. However, these measures fail to establish correspondence of the occluded pixels. In order to outperform the performance in the occluded pixels an modification of the weighted normalized cross correlation (WNCC) algorithm based on fuzzy relations of fuzzy data is used for the stereo matching process.

Authors of [21] present an modification of the Zitnick and Kanade stereo matching algorithm [22]. The authors propose the use of a balanced correlation window. In addition, the use of a new fuzzy factor in the calculation of initial matching values which expresses the possibility of matching between two pixels and a new iterative function for the refinement of the initial matching values are presented. Experimental results are evaluated on synthetic and real images and a comparison of the results regarding to other algorithms reported in the literature, using the ground truth data supplied by the University of Tsukuba, is presented.

Finally, in [23] an FPGA module suitable for real-time disparity map computation is presented. This enables a hardware-based fuzzy inference system parallel-pipelined design, for the overall module, implemented on a single FPGA device with a operating frequency of 138 MHz. This provides disparity map computation at a rate of nearly 440 frames per second, for a input stereo pair with a disparity range of 80 pixels and 640×480 pixel resolution. The proposed method allows high speed processing, enabling a suitable module for real-time stereo vision applications.

## 3. The proposed method

Stereo matching algorithms suitable for real-time processing have been studied by several authors, **section** 2.1. However, most of these algorithms possess high mathematical complexity; therefore, large FPGA devices are required in order implement them. On the other hand, although some approaches based on fuzzy logic have been reported in the literature **section** 2.2, the algorithms described in [20, 22] presented high mathematical complexity and do not allow to be implemented in dedicated hardware for real-time processing such as FPGA devices. In case of the FPGA architecture in [23] possesses high accuracy and high speed processing. However, the developed module maintains high hardware resource consumption.

The main objective in this research is to develop an FPGA stereo matching unit suitable for real-time processing. The disparity maps are computed at a rate of 76 frames per second for input stereo pairs of 1280×1024 pixel resolution and a maximum expected disparity ($d_{\max}$) equal to 15. In order to reach high accuracy level the use of multiple similarity parameters and a fuzzy approach is proposed. Furthermore, to obtain high speed processing, the proposed method was implemented in an FPGA device. On the other hand, in order to maintain low hardware resources consumption it is proposed the use of new strategies such as: a fuzzy inference system based on point-slope equations and the use of storage vectors. Finally, to obtain an appropriate configuration regarding to the environment setting of the input stereo pair, the developed FPGA architecture allows to set up the exposition of the cameras and allows scalability for different levels of maximum disparity of simple and systematic form. This allows that the resulting hardware can be applied to a wide range of applications of real-time stereo vision.

A general block diagram of the proposed method is shown in the **Figure** 1, the data obtained through a stereo camera is processed to the cadence of video and it is simultaneously computed three similarity parameters inherent to each pixel. 1 - correlation obtained via the Sum of absolute differences (SAD) algorithm [24–27]. 2 - similarity of the distance in pixels to the closer left edge. 3 - similarity of the distance in pixels the closer right edge. Later, these three parameters are submited to a fuzzy inference system which assigns fuzzy-similarity values for the zero up to $d_{\max}$ disparity levels. Finally, the value of disparity is assigned as the index that maximizes the fuzzy-similarity value.

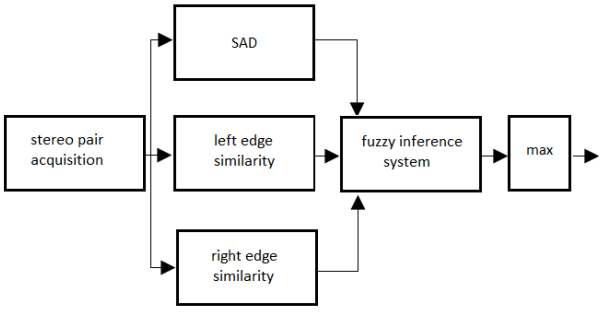



Figure 1: Block Diagram of the proposed method.

### 3.1. Correlation computation

In majority of area-based algorithms, a rectangular vicinity centered on a reference pixel in one of the images from a stereo pair is compared with similar vicinities for some pixels in the same raster line of the other image. Vicinities are called correlation windows and can be compared using a correlation-based measure such as SAD, **equation** 1; where $2w + 1$ is the size of the correlation window centered in the pixel located in the $(x, y)$ position. $I_l$, $I_r$ are the grayscale values of the pixels in the images left and right respectively and $z$ takes values from 0 up to $d_{\max}$.

$$\mathrm{Crl}(x, y, z) = \sum_{u=-w, v=-w}^{u=w, v=w} |I_l(x + u, y + u) - I_r(x + u + z, y + u)| \tag{1}$$

### 3.2. Similarity of the edge distances

The euclidean distance between each pixel with coordenates $I_l(x, y_1)$ or $I_r(x, y_1)$ and the closer left edge $(d_1)$, considering images of $X \times Y$ resolution and a maximum level of expected disparity equal to $d_{\max}$ is computed via **equations** 2 - 3; where $\beta$ represents the value of threshold to determine an edge. $\delta$ is defined as $l$ or $r$ for the left or right images, respectively. $z$ ranges from 0 up to $d_{\max}$. $y_1$ ranges from 1 up to $Y - d_{\max}$ and $g_1$ is defined by **equation** 4.

$$d_1(x, y_1, z) = \begin{cases} l = 0, & k_1(x, y_1, z) < \beta \\ l = l + 1, & k_1(x, y_1, z) > \beta \end{cases} \tag{2}$$

$$k_1(x, y_1, z) = |I_\delta(x, y_1 + z) - I_\delta(x, y_1 - g_1 + z)| \tag{3}$$

$$g_1 = \begin{cases} 1, & y > 1 \\ 0, & \text{otherwise} \end{cases} \tag{4}$$

In **Figure** 2 the left image of the Venus scene is shown. On the other hand, **Figure** 3 shown the distance to the closer left edge for all the pixels that integrate the scene. The darker values represent small distances whilst the brighter values represent great distances.

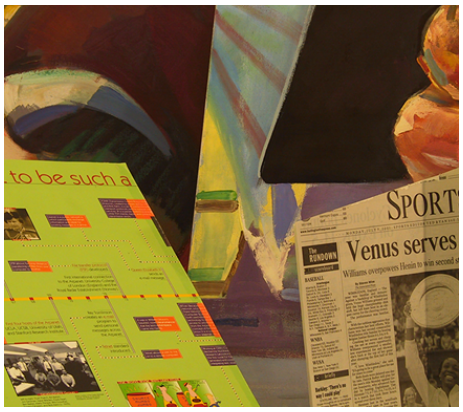



Figure 2: Venus scene, left image.

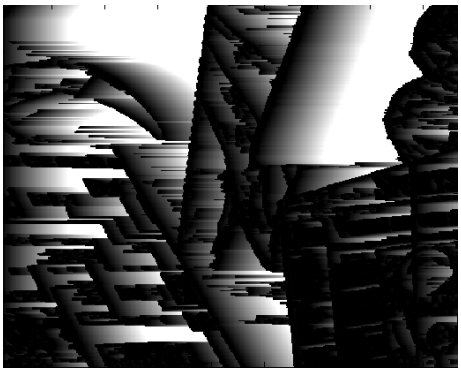


Figure 3: Venus scene, distance to the closer left edge.

The euclidean distance between each pixel with coordenates $I_l(x, y_1)$ or $I_r(x, y_1)$ and the closer right edge $(d_2)$, considering images of $X \times Y$ resolution and a maximum level of expected disparity equal to $d_{\max}$ is computed via **equations** 5 - 6; where $\beta$ represents the value of threshold to determine an edge. $\delta$ is defined as $l$ or $r$ for the left or right images, respectively. $z$ ranges from 0 up to $d_{\max}$. $y_1$ ranges from 1 up to $Y - d_{\max}$ and $g_1$ is defined by **equation** 7. In **Figure** 4 the distance to the closer right edge for all the pixels that integrate the Venus scene is shown.

$$d_2(x, y_1, z) = \begin{cases} l = 0, & k_2(x, y_1, z) < \beta \\ l = l + 1, & k_2(x, y_1, z) > \beta \end{cases} \tag{5}$$

$$k_2(x, y_1, z) = |I_\delta(x, y_1 + z) - I_\delta(x, y_1 + g_2 + z)| \tag{6}$$

$$g_2 = \begin{cases} 1, & y < X \\ 0, & \text{otherwise} \end{cases} \tag{7}$$

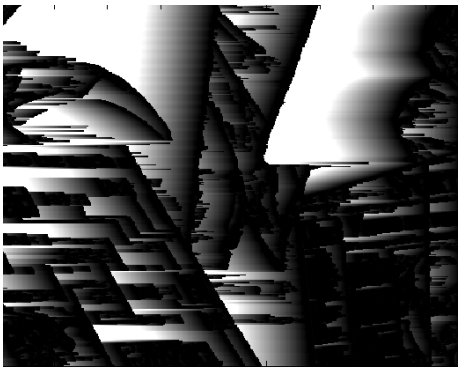


Figure 4: Venus scene, distance to the closer right edge.

In order to determine the similarity between the distances to the edges $d_l$, $d_r$ of the left and right images, $I_l$ and $I_r$, it is proposed the use of the equation 8; where $k$ is defined as $\{l, r\}$ and $z$ ranged from 0 up to $d_{\max}$.

$$D_k(x, y, z) = |I_l(d_k(x, y, 0)) - I_r(d_k(x, y, z))| \tag{8}$$

### 3.3. Fuzzy Inference System

The three similarity parameters, correlation (Crl) and the edge distances similarities $(D_l, D_r)$, ineherent to each $(x, y)$ pixel for each $z$ level of disparity, are submited to a fuzzy inference system that determines a fuzzy-similarity value. In the proposed fuzzy inference system, input membership functions for Crl, $D_1$ and $D_2$ are used. These functions determine the degree of truth to the fuzzy sets **good , medium** and **bad**, that represent the degree of similarity between the pixels of the left and right images, being **good** a high value of similarity, **bad** a low value of similarity and **medium** an intermediate value between high and low similarity.

Due to the linearity of the input data it is proposed to use triangular type functions with binary based slopes. Functions of this type allow to simplify the products and quotients to an shift operation over registers. This will be useful when implementing the proposed method in FPGA devices, this allow to reduce the hardware resources demand and maintaining low mathematical complexity. The input membership functions of for the **good, medium** and **bad** sets were defined as shown in **equations** 9 - 11; where $\alpha, \beta$ and $\gamma$ are equal to 0, 128 and 255 respectively. $h$ corresponds to the similarity parameters values.

$$\mu_{\text{good}}(x,y) = \begin{cases} 0, & \gamma \leq h \\ \frac{h-\gamma}{\alpha-\gamma} \cdot (255), & \alpha < h < \gamma \\ 255, & h \leq \alpha \end{cases} \tag{9}$$

$$\mu_{\text{medium}}(x,y) = \begin{cases} \frac{h-\alpha}{\beta-\alpha} \cdot (255), & \alpha < h < \beta \\ \frac{h-\gamma}{\beta-\gamma} \cdot (255), & \beta \leq h < \gamma \\ 0, & \text{otherwise} \end{cases} \tag{10}$$

$$\mu_{\text{bad}}(x,y) = \begin{cases} 0, & h \leq \alpha \\ \frac{h-\alpha}{\gamma-\alpha} \cdot (255), & \alpha < h < \gamma \\ 255, & \gamma \leq h \end{cases} \tag{11}$$

In the proposed method the output with the larger numerical value regarding to the diferent $z$ disparity levels. In order to assign output values three fuzzy output sets are proposed **good, medium** and **bad**. Where **good** represents the major numerical output values, descending to the values pertaining to the **medium** set and finally the values from the **bad** set that represents the minimum numerical values that can define the output. The degrees of truth to these sets are determined via the functions $\psi_{\text{good}}$, $\psi_{\text{medium}}$ and $\psi_{\text{bad}}$, **equations** 12 - 14; where $\alpha, \beta, \gamma$, $\rho$ and $\sigma$ are equal to 0, 128, 255, 96 and 160 respectively.

$$\psi_{\text{good}}(x,y) = \begin{cases} 0, & \gamma \leq h \\ \frac{h-\gamma}{\alpha-\gamma} \cdot (255), & \alpha < h < \gamma \\ 255, & h \leq \alpha \end{cases} \tag{12}$$

$$\psi_{\text{medium}}(x,y) = \begin{cases} \frac{h-\rho}{\beta-\rho} \cdot (255), & \rho < h < \beta \\ \frac{h-\sigma}{\beta-\sigma} \cdot (255), & \beta \leq h < \sigma \\ 0, & \text{otherwise} \end{cases} \tag{13}$$

$$\psi_{\text{bad}}(x,y) = \begin{cases} 0, & h \leq \alpha \\ \frac{h-\alpha}{\gamma-\alpha} \cdot (255), & \alpha < h < \gamma \\ 255, & \gamma \leq h \end{cases} \tag{14}$$

To relating the input sets to the output sets, the use of rules IF-THEN type and the AND operator of Zadet is proposed [28]. The design rules are shown in the Table 1. In order to use a fuzzy inference process with low hardware resource consumption, it is proposed the following fuzzy inference mechanism. First, the precedent of each rule is computed replacing the numerical values of the $Crl$, $D_1$ and $D_2$ inputs in **equations** 9-11; where $\varepsilon_n(x,y)$ is the precedent of the nth rule and $\mu_1(x,y), \mu_2(x,y)$ are both fuzzy sets included in each rule (Table 1). On the other hand, the consequent to each rule is determined as shown in **equation** 16; where $\vartheta_n(x,y)$ is the consequent of the nth rule. inp is the input value for the corresponding similarity parameter and $f_s$ is an point-slope type equation that describes a linear segment of the output membership functions, **equations** 12-14. In the column $f_r$ of the **Table** 1 the assignation of functions $f_s$ for each rule is shown.

$$\varepsilon_n(x,y) = \min\left(\mu_1(x,y), \mu_2(x,y)\right) \tag{15}$$

$$\vartheta_n(x,y) = f_s(\varepsilon_n(x,y)) \tag{16}$$

$$f_1(\varepsilon_n(x,y)) = \varepsilon_n(x,y) \tag{17}$$

$$f_2(\varepsilon_n(x,y)) = -\varepsilon_n(x,y) + 255 \tag{18}$$

$$f_3(\varepsilon_n(x,y)) = \begin{cases} (\varepsilon_n(x,y) - 768)/8, & \text{inp} < 128 \\ (-\varepsilon_n(x,y) + 1280)/8, & \text{otherwise} \end{cases} \tag{19}$$

Table 1: Bank of rules of the proposed fuzzy inference system

| **Rule number** | $Crl$ | $D_1$ | $D_2$ | **Output** | $f_s$ |
|---|---|---|---|---|---|
| 1 | *good* | *good* | *none* | *good* | $f_s = f_1$ |
| 2 | *good* | *medium* | *none* | *good* | $f_s = f_1$ |
| 3 | *good* | *bad* | *none* | *medium* | $f_s = f_3$ |
| 4 | *medium* | *good* | *none* | *medium* | $f_s = f_3$ |
| 5 | *medium* | *medium* | *none* | *bad* | $f_s = f_2$ |
| 6 | *medium* | *bad* | *none* | *medium* | $f_s = f_3$ |
| 7 | *bad* | *good* | *none* | *medium* | $f_s = f_3$ |
| 8 | *bad* | *medium* | *none* | *medium* | $f_s = f_3$ |
| 9 | *bad* | *bad* | *none* | *bad* | $f_s = f_2$ |
| 10 | *good* | *none* | *good* | *good* | $f_s = f_1$ |
| 11 | *good* | *none* | *medium* | *medium* | $f_s = f_3$ |
| 12 | *good* | *none* | *bad* | *medium* | $f_s = f_3$ |
| 13 | *medium* | *none* | *good* | *bad* | $f_s = f_2$ |
| 14 | *medium* | *none* | *medium* | *medium* | $f_s = f_3$ |
| 15 | *medium* | *none* | *bad* | *medium* | $f_s = f_3$ |
| 16 | *bad* | *none* | *good* | *medium* | $f_s = f_3$ |
| 17 | *bad* | *none* | *medium* | *bad* | $f_s = f_2$ |
| 18 | *bad* | *none* | *bad* | *bad* | $f_s = f_2$ |

Finally, the result of the fuzzy inference is determined using the Mean-Max method, **equation** 20; where $\vartheta_{m1}(x,y), \vartheta_{m2}(x,y)$ are the consequence of the two rules with higher implication in the fuzzy inference process while the corresponding disparity value is determined via the **equation** 21.

$$\text{fuzzy\_inference}(x,y,z) = (\vartheta_{m1}(x,y) + \vartheta_{m2}(x,y))/2 \tag{20}$$

$$\text{disparity}(x,y) = \arg\max_z \text{ fuzzy\_inference}(x,y,z) \tag{21}$$

### 3.4. Output mapping

In order to turn the disparity values defined by values from 0 up to ($d_{\max}$) to RGB values with 10 bits of depth, it is proposed to applying the **equation** 22; where $d_{\max}$ is the maximum expected disparity value $\forall\ \eta\ \epsilon\ \{1, 2, 3\}$.

$$\text{disparity}_{\text{mapped}}(x, y, \eta) = \text{disparity}(x, y) * 1023/d_{\max} \quad (22)$$

### 3.5. Computational complexity

In order to explain the computational complexity of the proposed algorithm, first, the SAD computational complexity is analyzed. In this case the computational complexity is defined as following: $O_{\text{SAD}}(M\ S\ D/d')$; where $M$ is the size of the input stereo pair. $S$ is the size of the correlation window. $D$ is the maximum expected disparity and $d'$ is the increment regarding to the disparity values. Like SAD, the proposed algorithm possesses a computational complexity defined in the same terms, **Table** 3. i.e. $O_{\text{SAD}} = O_{\text{propoed}} = O(M\ S\ D/d')$. Based on the high efficient of to the SAD algorithm and considering that his computational complexity is equal to the complexity of the proposed algorithm. It is possible to affirm the high efficient of the proposed method.

On the other hand, although the proposed method possesses mathematical complexity similar to SAD, due to edges extraction, similarity computation stage and the fuzzy inference process, the runtime is increased. In **Table** 2 the runtime of the SAD algorithm considering different synthetic stereo pairs and different values for the maximum expected disparity compared with the proposed method runtime considering similar settings is presented. As can be seen, in all the cases an increment of the runtime is observed, near to 10 times. However, most of the new runtime corresponds to the fuzzy inference process. In MatLab this is an iterative process that computes in sequential form the degree of truth for all the input fuzzy sets. Then evaluates in sequential form all the fuzzy rules and assigns the corresponding antecedent and consequent values. Finally the fuzzy similarity value is computed. However, this process could be parallelized when the proposed method is implemented in dedicated hardware such as FPGA devices that enables to reach real-time processing. Furthermore, considering the proposed fuzzy inference method any FPGA architecture must maintain low hardware resources consumption.

Table 2: Runtime for the proposed method

| **Scene/Algorithm** | **Window size** $(2w+1)$ | | | | | |
|---|---|---|---|---|---|---|
| | 3 | 5 | 7 | 9 | 11 | 13 |
| Tsukuba/SAD | 7.48 | 9.32 | 11.5 | 14.2 | 16.9 | 21.2 |
| Venus/SAD | 10.2 | 13.1 | 17.0 | 22.5 | 28.6 | 36.1 |
| Teddy/SAD | 16.7 | 23.5 | 33.5 | 48.0 | 63.9 | 83.0 |
| Cones/SAD | 16.7 | 23.5 | 33.5 | 48.0 | 63.9 | 83.0 |
| Tsukuba/proposed | 79.5 | 99.3 | 123 | 154 | 193 | 241 |
| Venus/proposed | 108 | 135 | 168 | 210 | 262 | 327 |
| Teddy/proposed | 177 | 221 | 276 | 345 | 431 | 561 |
| Cones/proposed | 177 | 221 | 276 | 345 | 431 | 561 |

*All the runtimes are measured in seconds and were obtained via MatLab implementations

Finally, in **Table** 3 the detailed pseudocode for the proposed method is shown. As can be seen this consist into several recursive and sequential operations, such as edge extraction stage, edge similarity computation, fuzzy inference process and so on, characteristics that allow parallel-pipelined design suitable for FPGA devices.

Table 3: Pseudo code for the proposed stereo matching algorithm

**Parameter definition:**

$H$: The size of an image $I$ ($H = X_{\text{resolution}} \times Y_{\text{resolution}}$)

$W$: The size of a correlation window ($W = (2 \times w + 1)^2$)

$S$: The maximum expected disparity ($S = S_m$)

$s'$: Increase of disparity ($s' = 1$)

**Algorithm:**

**Complexity:** $O(H\ W\ S/s')$

**For** all pixels $p(x, y)$ which satisfy $x >= w, y >= w$ and $x <= X_{\text{resolution}} - w - S, y <= Y_{\text{resolution}} - w$

  **For** $z = 1$ with increments equal to $s'$ up to $S$

    1: Compute the Crl$(x, y, z)$ value, **equation** 1

  **End**

**End**

**For** all pixels $p(x, y)$ which satisfy $x >= w, y >= w$ and $x <= X_{\text{resolution}} - w - S, y <= Y_{\text{resolution}} - w$

  **For** $z = 1$ with increments equal to $s'$ up to $S$

    2: Compute the $d_l(x, y, z)$ value, **equation** 2, $\delta$=$l$

    3: Compute the $d_l(x, y, z)$ value, **equation** 2, $\delta$=$r$

    4: Compute the $D_k(x, y, z)$ value, **equation** 8, $k = l$

  **End**

**End**

**For** all pixels $p(x, y)$ which satisfy $x >= w, y >= w$ and $x <= X_{\text{resolution}} - w - S, y <= Y_{\text{resolution}} - w$

  **For** $z = 1$ with increments equal to $s'$ up to $S$

    5: Compute the $d_r(x, y, z)$ value, **equation** 5, $\delta$=$l$

    6: Compute the $d_r(x, y, z)$ value, **equation** 5, $\delta$=$r$

    7: Compute the $D_k(x, y, z)$ value, **equation** 8, $k = r$

  **End**

**End**

**For** all pixels $p(x, y)$ which satisfy $x >= w, y >= w$ and $x <= X_{\text{resolution}} - w - S, y <= Y_{\text{resolution}} - w$

  **For** $z = 1$ with increments equal to $s'$ up to $S$

    **For** $n = 1$ with increments equal to 1 up to 18

      8: Compute the $\varepsilon_n(x, y)$ value, **equation** 15

      9: Compute the $\vartheta_n(x, y)$ value, **equation** 16

    **End**

    10: Compute the fuzzy_inference$(x, y, z)$ value, **equation** 20

  **End**

  11: Compute the disparity$(x, y)$ value, **equation** 21

**End**

**For** all pixels $p(x, y)$ which satisfy $x >= w, y >= w$ and $x <= X_{\text{resolution}} - w - S, y <= Y_{\text{resolution}} - w$

  12: Compute the $\text{disparity}_{\text{mapped}}(x, y, \eta)$ value, **equation** 22

**End**

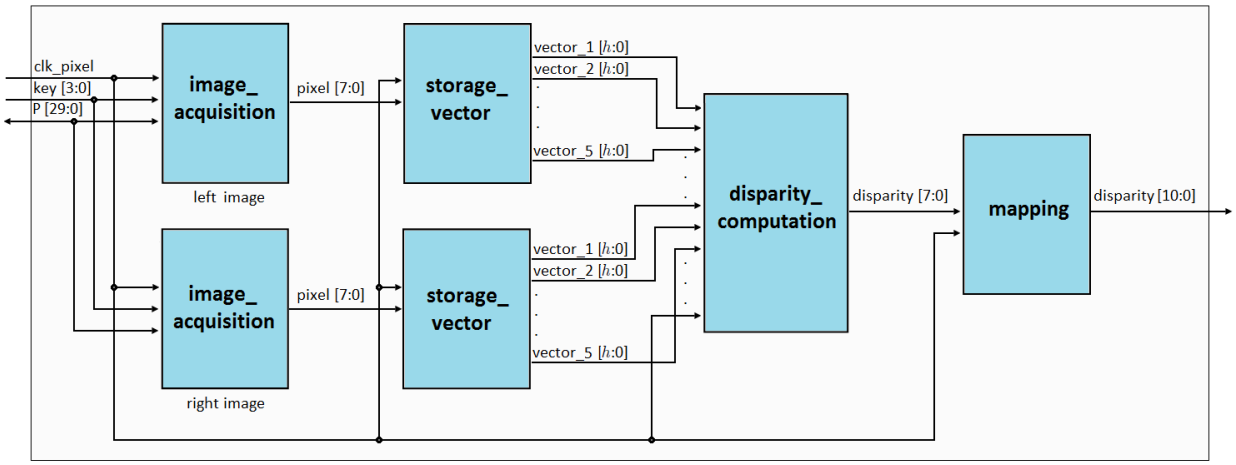



Figure 5: General diagram of the developed FPGA architecture

## 4. FPGA architecture

In **Fig.** 5, an overview of the developed FPGA architecture is shown. This architecture has three inputs. `clk_pixel` as the pixel rate of the input stereo pairs. `key [3:0]` as as control signals for the left and right cameras and `P [29:0]` as input/output bus for the cameras. On the other hand, the developed architecture has one output, `disparity [10:0]`, corresponding to disparity value for the selected pixel. The developed FPGA architecture was set to process input stereo pairs of $1200 \times 1024$ pixel resolution, and considering a maximum expected disparity equal to 15. Its general behavior can be described as following: first, the **image_acquisition** modules capture stereo video streams. Then, the **storage_vector** modules store gray scale values of pixels contained in 5 horizontal lines for both left and right images of input stereo pair. Later, the disparity value are computed via **disparity_computation** module. Finally, the **mapping** convert the final disparity value to grayscale values of 10 bits of depth. In the following subsections the architecture of all the individual modules is shown in detail.

### *4.1. The **image_acquisition** module*

In order to acquire input stereo pairs, CCD sensors connected in a TRDB DC2 board is used. This board provides stereo pairs of 1280×1024 pixel resolution in RGB scale. In most of the FPGA stereo matching implementations reported in the literature, grayscale values with 8 bits of depth are used. However, the TRDB DC2 board provided images in RBG format with 10 bits of depth. Although the developed architecture could be configured to process 10 bits of depth, this increases the hardware resource consumption. In order to perform appropriate comparisons regarding to other methods, only the most significant 8 bits of the data provided by the TRDB DC2 board are used. On the other hand, in order to determine the grayscale value for the input images, the value of the green channel is used as grayscale value.

In **Figure** 6 the detailed architecture of the **image_acquisition** module is shown. The **settings** module send instructions regarding to the exposition and operation mode to the TRDB DC2 board, `P [29:0]`. The value of the exposition is defined via the `Q [7:0]` input while the operation mode is defied by the `G [3:0]`, **Table** 4. The **frame_capture** module consider the user settings `S [15:0]` and captures frames of 1280×1024 pixel resolution, obtaining at the `D [7:0]` output a binary chain with the RGB value form one pixel of the captured frame while the `x [7:0]`, `y [7:0]` outputs consist on the indexes of each pixel. Finally, **raw_to_grayscale** enables to obtain grayscale values with 8 bits of depth, `P [7:0]` output.

Table 4: Operation specifications for the TRDB_DC2 board

| Name | Description |
|---|---|
| G 0 | Reset the frame capture |
| G 1 | Assign the exposure value |
| G 2 | Pause the frame capture |
| G 3 | Continuous frame capture |

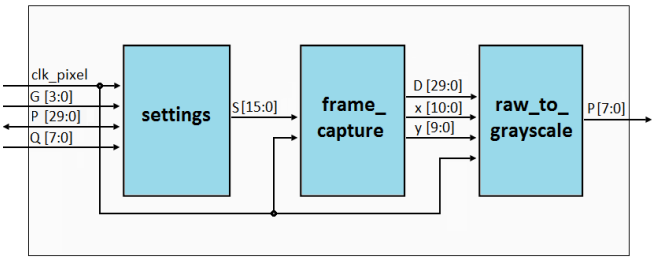



Figure 6: FPGA architecture for the **stereo_pair_acquisition** module.

### *4.2. The **storage_vector** module*

In order to store necessary data for the disparity computation, the use of the **storage_vector** module is proposed. This module consists into five logic vectors. The size of four vector is equal to `horizontal_resolution`×`bits_per_pixel` − 1 while one vector size is defined as $d_{\max} \times$ `bits_per_pixel` − 1.

Although the proposed module possesses similar behavior with respect to a shift register unit, this allows to read multiple data in one clock cycle. In general, when a image line begins, the gray scale value of the pixel with coordinate (1) is stored in index [7:0] of one storage vector, in the following clock cycle, this value is moved to index [15:8] and the gray scale value of the pixel with coordinate (2) is stored in index [7:0]. A similar process is repeated for all the pixels that integrate the line. In **Figure** 7, behavior of **storage_vector** module with settings as follows: number of lines to process = 5, horizontal resolution = 1200.

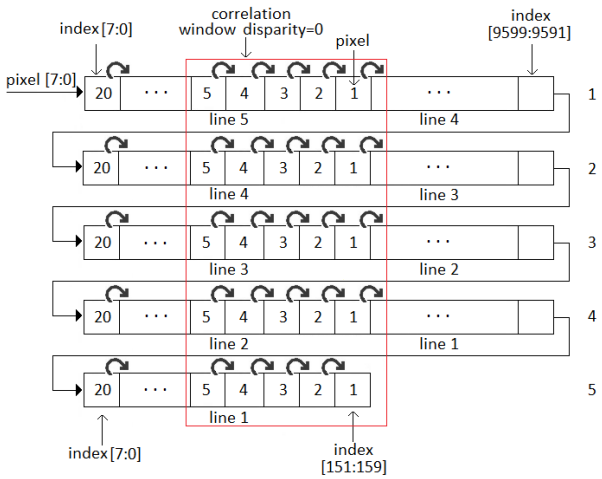



Figure 7: Behavior of the **storage_vector** module

### 4.3. The ***disparity_computation*** module

In **Figure** 8 the detailed architecture of the **image_acquisition** module is shown. An pixel-parallel window-parallel architecture was designed. Using the appropriate indexes for the **storage_vector** is possible to process the image to the video cadence, giving as result a disparity map of $(X - w)*(Y - w)$ pixels of resolution, where $X$, $Y$ are the values of resolution of the input frames and $2w + 1$ is the dimension of the correlation window used. In the following subsections, details regarding to the operation of the module **disparity_computation** is shown.

#### 4.3.1. The ***similarity_parameter*** module

The **similarity_ parameter** module calculates the similarity parameters necessary to estimate the disparity value for one particular pixel, $\mathrm{Crl}(x, y, z)$, $D_l$, $D_r$. This module consists into three sub-modules implemented in parallel form. Furthermore, a 5×5 correlation window and maximum expected disparity equal to 15 are used. In order to compute the first parameter $(\mathrm{Crl}(x, y, z))$; the first sub-modules performs the absolute difference between all pixels in the correlation window in parallel form. Then, all absolute difference values are summed.

The second sub-module computes the second similarity parameter $D_l$. First, this sub-module applies the **equations** 3 and 4 and computes the $k_1$ value. The **Figure** 9 shows the position of the pixels used in this stage; by maintaining this position it is possible to compute the $d_l$ value at the video cadence. In addition, this figure shows a scenario in which the storage process for the five lines of the first correlation window have been completed; as can be seen the third vector correspond to the central line of the correlation. Due to shift behavior it is possible to compute the $k_1$ value at video cadence. After the $k_l$ value is computed , it is determines the $d_1$ value by assigning to $\beta$ (**equation** 2) a constant value equal to 32. Finally, the similarity value between the edges of the left and right images is computed via **equation** 8.

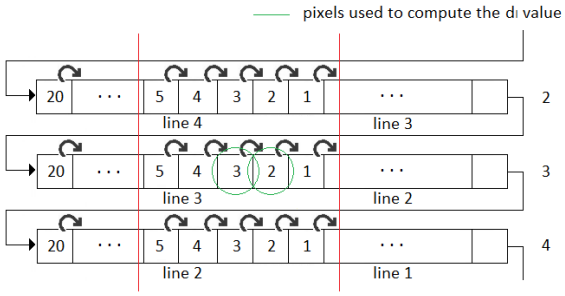



Figure 9: Behavior of the **storage_vector** module

Finally, the third sub-module computes the third similarity parameter, $D_r$. This sub-module possesses a similar behavior to the $D_l$ computation sub-module. First, it is applied the **equations** 6 and 7 and it is computed the $k_r$ value. After the $k_r$ value is computed , it is determines the $d_r$ value by assigning to $\beta$ (**equation** 5) a constant value equal to 32. Finally, the similarity value between the edges of the left and right images is computed via **equation** 8. However, the **storage_vector** module, **Figure** 7 only allows to determine the left edge distance values. In order to compute the right edge distance values it is proposed the use of the **right_direction_vector** module.

#### 4.3.2. The ***right_direction_vector*** module

This module consist into three stages. First, it is determined when all the pixels from one line of the input frame are stored in the number 1 vector of the **storage_vector** module, **Figure** 7. i.e., it is determined when the storage process for all the pixels of the line 1 or 2 or 3 and so on is completed. Then, in the second stage the values contained in the number 1 vector of the **storage_vector** module are copied and placed in the $k_r$ vector, **Figure** 11-(a). The $k_r$ vector possesses a similar behavior to the vectors of the **storage_vector** module, however, the order in the data of the $k_r$ are inverted. Using the $k_r$ vector it is possible to obtain the $D_r$ value. Finally, in the third stage the $d_r$ values are placed in the $d_r$ vector. The behavior of this vector is shown in the **Figure** 11-(b).

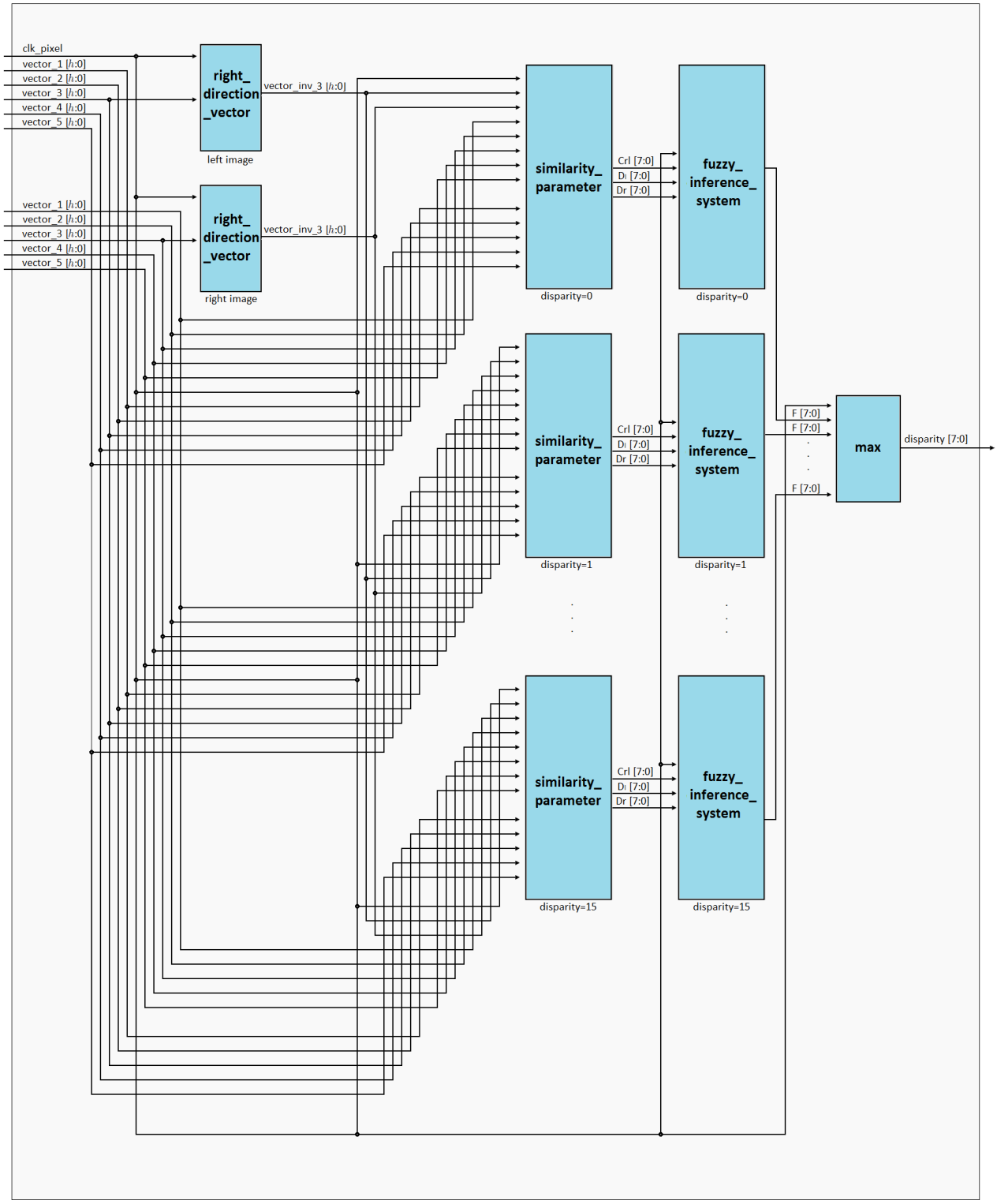



Figure 8: FPGA architecture of the **disparity_computation** module

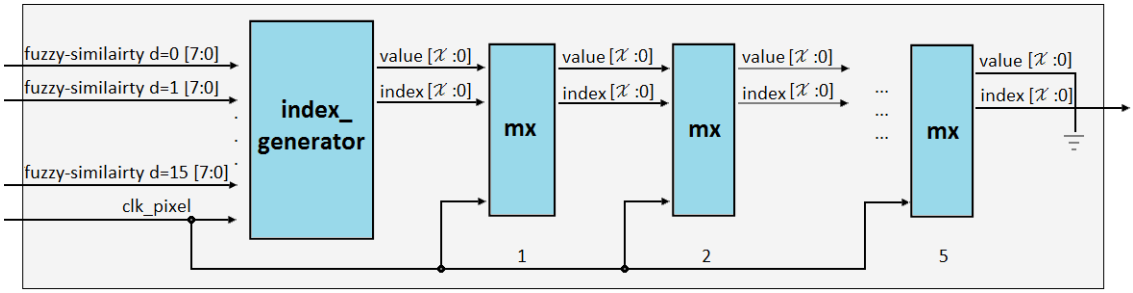
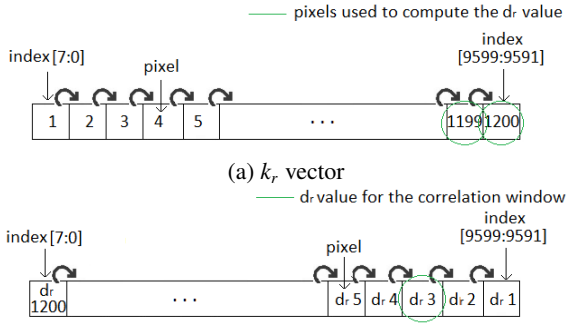



Figure 10: Behavior of the **storage_vector** module




(a) $k_r$ vector

(b) $d_r$ vector

Figure 11: Pixels used to compute the $D_l$ value

#### 4.3.3. The ***fuzzy_inference_system*** *module*

This module consist into four different sub-modules implemented in sequential form. The first sub-module conducts the fuzzification process for the three input parameters (Crl, $D_l$ and $D_r$). With the purpose of reserving the logical elements (LEs) of the FPGA for the defuzzification stage, the **equations** 9-11 were evaluated with values of 0 up to 255 with increases equal to 1, the obtained values were integrated in a **CASE** structure. This structure assigns the degrees of truth to the fuzzy input sets **good**, **medium** and **bad** for the three similarity parameters. Later, in the second sub-module the 18 rules that integrate the proposed bank of rules are evaluated in parallel form, the antecedent and consequent regarding to each rule are calculated according to the **equations** 15-16. Then, the 18 calculated consequent enter to the **max** third module that determines the two rules with higher implication in the inference process. Finally, the fourth module computes the arithmetic mean of its two inputs, defuzzification process. i.e. computes the fuzzy-similatity value.

#### 4.3.4. The ***max*** *module*

In order to reach an appropriate propagation of the processed data, the use of the **max** actor is proposed, **Figure** 10. It consists into one **index_generator** module and 5 **maximum** modules implemented in sequential form. First, the **index_generator** module assigns the corresponding indexes to all the fuzzy-similarity values values from, then, the **maximum**$_1$ module, receives all the fuzzy-similarity values and their indexes. Afterwards, this module determines the maximum values for correlation values, which are sorted by pairs with unrepeated correlation values for any pair; the minimum correlation values and their indexes obtained here are placed in logic vectors of type `value [x:0]`, where $x = 16\times(d_{\max}+1)-1$, and `index [x:0]`, where $x = 8 \times (d_{\max} + 1) - 1$, respectively. This process is repeated in sequential form five times only one correlation value and its index are placed in the output vectors. At this time, the stored index are equal to the disparity value.

### 4.4. The ***mapping*** *module*

In order to display the calculated disparity maps a 4,3" LCD screen of 800x480 pixels of resolution of the terasIC brand was used. Due to the used screen operates with color values of 10 bits per pixel, the **mapping** module, **Fig.** 5, turns the disparity values, in this case velues ranged from 0 to 15, to grayscale values with 10 bits per pixel, **equation** 22. In order to reduce the hardware resources requirements the **equation** 22 was evaluated with disparity values from 0 up to 15 with increases equal to 1, the obtained values were integrated in a CASE structure which assigns disparity values with 10 bits per pixel.

## 5. Discussion and analysis of results

The architecture presented in Section 3 was implemented using a top-down approach. All modules were codified in Verilog and were simulated using post-synthesis simulations performed in ModelSim-Altera 6.6c in order to verify its functionality. Quartus II Web Edition version 10.1SP1 was used for the synthesis and FPGA implementation process. An FPGA Cyclone II EP2C35F672C6 embedded in an Altera DE2 development board was used. The hardware resource consumption for the developed FPGA architecture is shown in **Table** 5.

Table 5: Hardware resource consumption for the developed FPGA architecture.

| Resource | Demand |
|---|---|
| Total logic elements | 22,075 (66.89%) |
| Total pins | 161/475 (34%) |
| Total Memory Bits | 390,656 (81%) |
| Embedded multiplier elements | 0/70 (0%) |
| Total PLLs | 0/4 (0%) |

In order to evaluate the performance of the proposed method, the developed architecture was analyzed using different values for the $\beta$ parameter (**equations** 2, 5) and different sizes for the correlation windows. The tests were conducted using as test images the Tsukuba, Venus, Teddy and Cones scenes. $\beta$ was evaluated using the values $\{8, 16, 32, 64, 128\}$ while the window sizes used $(2w + 1)$ were $\{3, 5, 7, 9, 11, 13, 15\}$. In the **Figure** 12 is observed the behavior of the error obtained in the disparity maps generated for different values of $\beta$ considering $2w + 1 = 5$. On the other hand, in **Figure** 13 the behavior of the error obtained in the disparity maps generated for different values of $2w+1$ considering $\beta = 32$ is shown. To determine the number of erroneous pixels, the corresponding disparity maps were submited in the website for evaluation of stereo systems of the Middlebury University [29].

In the **Table** 6 it is presented quantitative results of the number of erroneous pixels obtained by the proposed method for a correlation window of 5x5 ($2w + 1 = 5$) and $\beta = 32$ compared with other methods reported in the Literature. Disparity maps have been compared using the method proposed in [30] using the following configuration of images: Tsukuba (384×288, $d_{\max} = 15$), Venus (434×383, $d_{\max} = 19$), Teddy (450×375, $d_{\max} = 59$),Cones (450×375, $d_{\max} = 59$). In order to process and to collect the data presented in the **Table** 6, the developed architecture was scaled to operate with 64 levels of disparity. In **Figure** 14 the generated disparity maps for the proposed algorithm are shown.

Table 6: Comparison between quantitative results for different real-time stereo matching algorithms.

| Method | Tsukuba | Venus | Teddy | Cones |
|---|---|---|---|---|
| SAD | 16.3% | 28.5% | 43.7% | 39.8% |
| [31] | 12.0% | 8.0% | - | - |
| [32] | 15.2% | 14.1% | - | - |
| [33] | 12.8% | 10.8% | 10.7% | - |
| [34] | 6.2% | 2.4% | 13.8% | 9.5% |
| [35] | 7.5% | 4.1% | 17.6% | 18.4% |
| [36] | 10.4% | 12.1% | 29.1% | 25.3% |
| [37] | 11.5% | 5.27% | 21.5% | 17.5% |
| Porposed | 6.84% | 6.07% | 27.3% | 22.4% |

By analyzing **Table** 6, due to use of geometrical features such as edge distances the proposed algorithm allows high accuracy level in homogeneous areas, therefore, the results of the proposed algorithm improved most of SAD, SSD or NCC-based algorithms reported in literature, [31–33, 36, 37]. In addition, similar to most of stereo matching algorithms in the literature, the proposed algorithm has high performance with small values of maximum disparity (Tsukuba, Venus scenes) while medium performance with large values of maximum disparity (Teddy, Cones scenes). On the other hand, the proposed algorithm performance is lower than the census based algorithms, [34, 35], however, census-based algorithms involve high mathematical complexity and the hardware resources requirements are high.

**Table** 7 presents comparisons of processing speed regarding to other real-time stereo matching algorithms reported in the literature. Due to the mathematical simplicity of the proposed algorithm, the developed architecture does not require complex arithmetical operations such as calculation of quotients and radicals (which require a high runtime), hence, it maintains high processing speed. When comparison of processing speed is conducted, **Table** 7, it is observed an increase with respect to other algorithms implemented in FPGA devices up to 78,725,120, minimum 14,417,920 pixels per second. In all cases except for the algorithm presented in [33],the processing speed for the proposed method is higher. However, considering the clock specifications for the algorithm described in [33], it is expected higher speed processing than the proposed algorithm. It is possible to affirm that if the same clock specifications are used, the proposed algorithm outperforms the speed processing for most of the FPGA-based stereo matching algorithms reported in the literature.

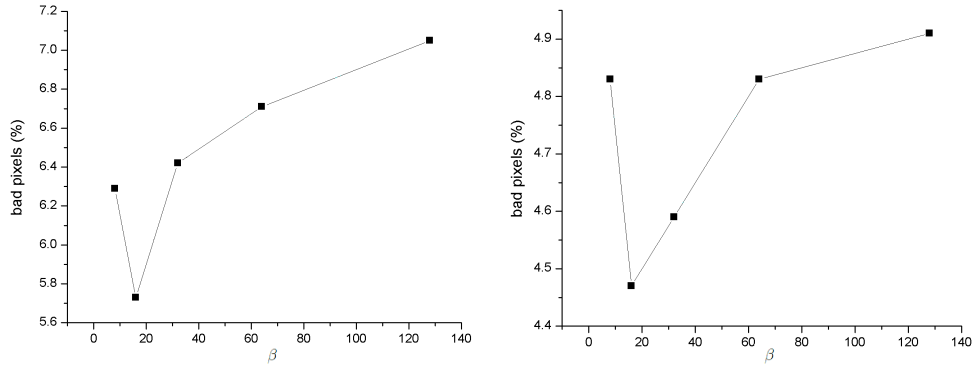
Table 7: Processing speed for differents real-time stereo matching algorithms

| Method | Resolution | Frames/s | Pixels/s | Clock |
|---|---|---|---|---|
| [33] | 1024×1024 | 102 | 106,954,752 | 441MHz. |
| [31] | 1280×1024 | 65 | 85,196,800 | 246MHz. |
| [34] | 320×240 | 573.7 | 44,060,160 | 3GHz. |
| [35] | 640×480 | 68 | 20,889,600 | 120MHz. |
| [36] | 1280×1024 | 50 | 65,536,000 | 109MHz. |
| [37] | 640×480 | 230 | 70,656,000 | 245MHz. |
| Proposed | 1280×1024 | 76 | 99,614,720 | 100MHz. |
| Proposed | 450×375 | 592 | 99,900,000 | 100MHz. |
| Proposed | 434×383 | 601 | 99,899,422 | 100MHz. |
| Proposed | 384×288 | 904 | 99,975,168 | 100MHz. |

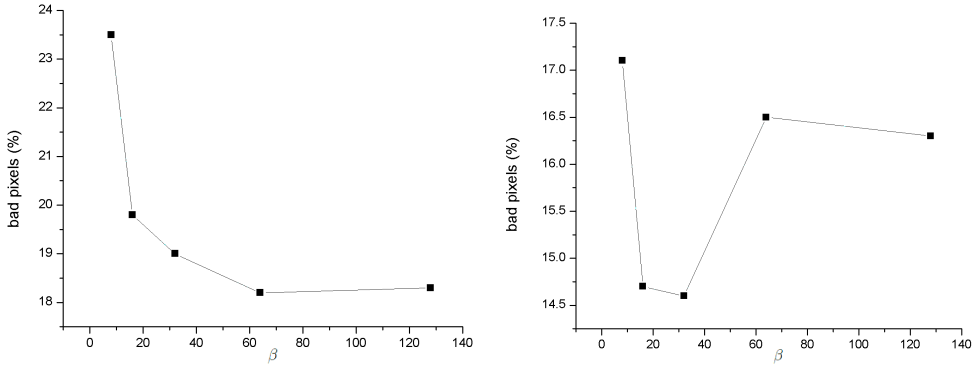
**Table** 5 presents a comparison of the use of hardware resource regarding to other FPGA implementations in the literature. By analyzing the **Figure** 8, the higher performance for the proposed algorithm can be reached using a 5×5 correlation window. It is the main characteristic and difference regarding to all others stereo matching algorithms in the literature whose performance is proportional to the correlation window size. In order to reach high performance, large correlation window sizes are required. However, the correlation window size required for the proposed method is small, therefore, it is possible to decrease the hardware resource consumption, see **Table** 8.

Table 8: Hardware resource consumption for different FPGA-based stereo matching algorithms

| Method | Logic Elements (LEs) | Memory bits (ALUTs) |
|---|---|---|
| [33] | 86,252 | 62,669 |
| [31] | 89,459 | 84,307 |
| [35] | 80,270 | 32 RAMs |
| [36] | 31,863 | 49,331 |
| [37] | 53,616 | 60,598 |
| Proposed | 22,075 | 48,832 |

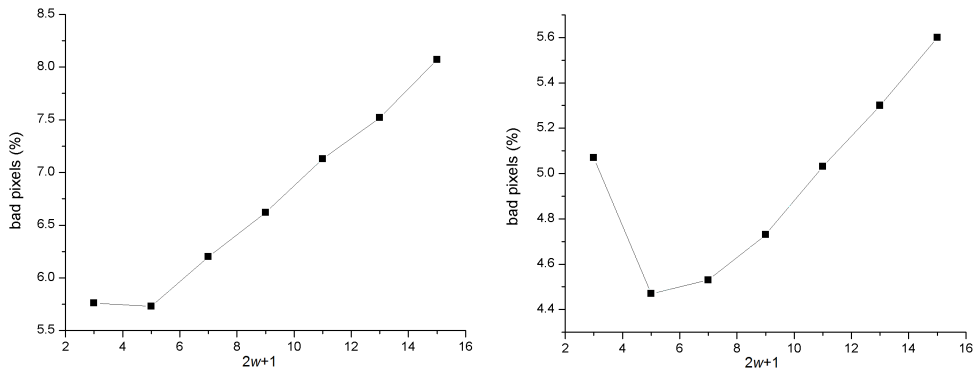



(a) Tsukuba scene (b) Venus scene

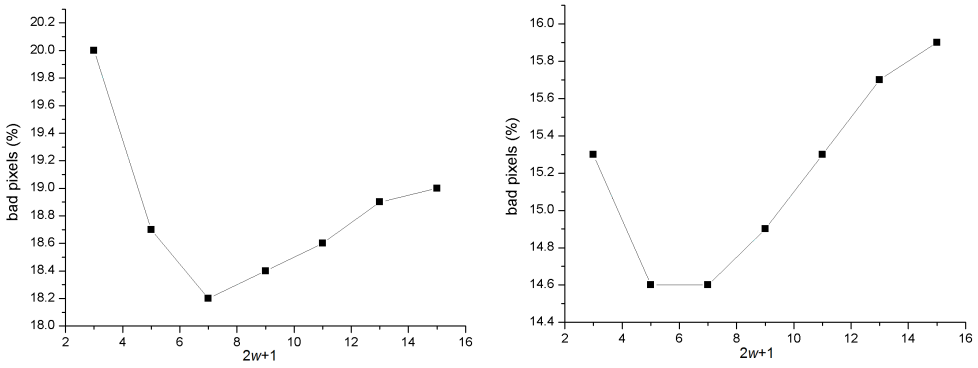



(a) Teddy scene (b) Cones scene

Figure 12: $\beta$ parameter behavior.

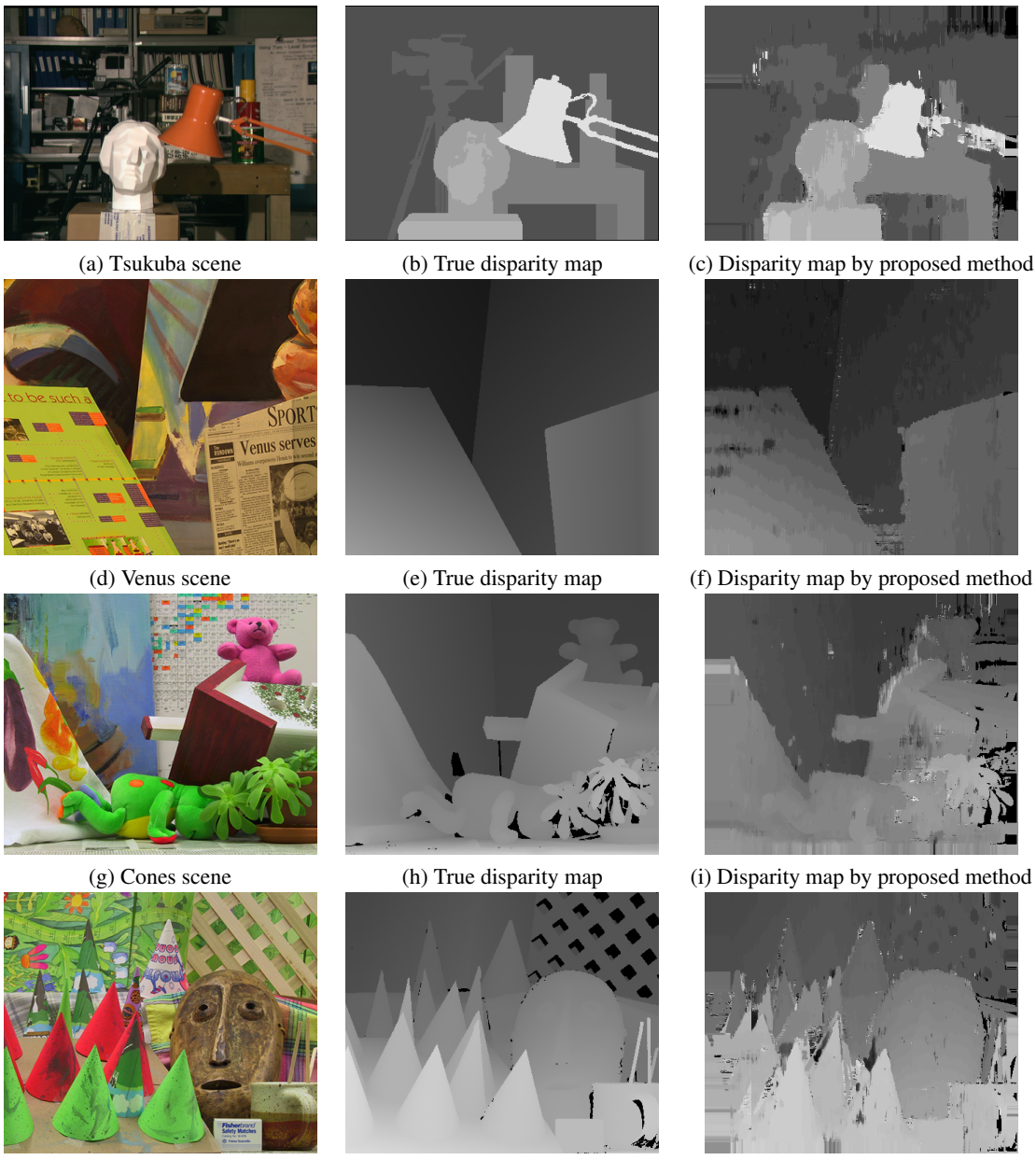



(a) Tsukuba scene (b) Venus scene




(a) Teddy scene (b) Cones scene

Figure 13: Behavior for different correlation window sizes.




(a) Tsukuba scene (b) True disparity map (c) Disparity map by proposed method

(d) Venus scene (e) True disparity map (f) Disparity map by proposed method

(g) Cones scene (h) True disparity map (i) Disparity map by proposed method

(j) Teddy scene (k) True disparity map (l) Disparity map by proposed method

Figure 14: Disparity maps generated for different test images.

Finally, in **Tables** 9 and 10 present a comparison of the use of hardware resource regarding to all the synthesized and simulated configurations of the developed FPGA architecture. as can bee seen the hardware consumption for the developed FPGA architecture is appropriate for the majority of the low gamma of FPGA devices such as the Cyclone II family of Altera or Spartan II family of Xilinx. Low hardware resources requirements is useful in autonomous applications such as robotic applications, where the use of small FPGA devices that implies relatively few hardware resources is needed. On the other hand, due to high speed processing, low hardware resources consumption and high accuracy level compared with most of the FPGA implementations in the literature, any real-time application wich involve real-time stereo matching algorithms will take advantages by applying the proposed algorithm.

Table 9: Logic elements (combinational functions and logic registers) consumption for different configurations of the developed FPGA architecture

| $d_{\max}$ / resolution | **15** | **31** | **63** |
|---|---|---|---|
| 384x288 | 21,967 LEs | 23,015 LEs | 24,871 LEs |
| 434x383 | 22,003 LEs | 23,621 LEs | 25,457 LEs |
| 450x375 | 22,035 LEs | 24,145 LEs | 26,013 LEs |
| 1280x1024 | 22,075 LEs | 24,450 LEs | 26,487 LEs |

Table 10: Memory bits consumption for different configurations of the developed FPGA architecture

| $d_{\max}$ / resolution | **15** | **31** | **63** |
|---|---|---|---|
| 384x288 | 20,160 bits | 20,296 bits | 20,552 bits |
| 434x383 | 21,760 bits | 21,869 bits | 22,152 bits |
| 450x375 | 22,270 bits | 22,408 bits | 22,664 bits |
| 1280x1024 | 48,832 bits | 48,968 bits | 49,224 bits |

## 6. Conclusions

In this article, an area-based algorithm suitable for real-time stereo matching using similarity measures and geometric features was presented. Geometric features such as edge sitantaces allow high accuracy level in homogeneous areas. Then, in order to relate the similarity measures with the geometrical features, the use of a fuzzy inference system is proposed. Furthermore, in order to maintain low mathematical complexity suitable for FPGA implementations, a fuzzy inference mechanism based in point-slope equations is proposed. The main difference and advantage for the proposed algorithm is the use of small correlation windows in order to reach high accuracy level. Small correlation windows allow higher speed processing and lower computational resources requirement in comparison with the large correlation windows used in most of the SAD-based stereo matching algorithms reported in the literature.

In order to improve its processing speed, the proposed algorithm was implemented in a FPGA device. The developed FPGA architecture outperforms most of the real-time stereo matching algorithms in the literature, allowing high accuracy level and enables both increasing the processing speed and decreases the hardware resources consumption.

Finally, an important characteristic of the presented architecture is the scalability permissible; all the modules in the developed FPGA architecture, allow to be adapted for processing larger correlation windows than the simulated and implemented correlation windows. On the other hand, the FPGA architecture enables to configure different levels of maximum expected disparity ($d_{\max}$), consequently, it is possible to configure the module for the computation of disparity maps with appropriate values to the environmental characteristics of the input video streams. This allows that the developed architecture can be applied to a wide range of applications of real-time stereo vision such as positioning systems for mobile robots and recognition, detection and so on.

## 7. Referencias


[1] E. Parrilla, J.-R. Torregrosa, J. Riera, and J.-L. Hueso, "Fuzzy control for obstacle detection in stereo video sequences," *Mathematical and Computer Modelling*, vol. 54, pp. 1813–1817, 01 2011.

[2] X. Rong, J. Huanyu, and Y. Yibin, "Recognition of clustered tomatoes based on binocular stereo vision," *Computers and Electronics in Agriculture*, vol. 106, pp. 75–90, 05 2014.

[3] R. Correal, G. Pajares, and J. Ruz, "Automatic expert system for 3d terrain reconstruction based on stereo vision and histagoram matching," *Expert Systems with Applications*, vol. 106, pp. 75–90, 09 2013.

[4] J. A. Delmerico, P. Davidb, and J. J. Corso, "Building facade detection, segmentation, and parameter estimation for mobile robot stereo vision," *Image and Vision Computing*, vol. 31, pp. 841–852, 08 2013.

[5] F. Åuligoj, B. Åekoranja, M. Åvaco, and B. Jerbi, "Object tracking with a multiagent robot system and a stereo vision camera," *Procedia Engineering*, vol. 69, pp. 968–973, 2014.

[6] L. Yang and N. Noguchi, "Human detection for a robot tractor using omni-directional stereo vision," *Computers and Electronics in Agriculture*, vol. 89, pp. 112–125, 08 2012.

[7] D. B. A. C. Roberto Marzotto, Paul Zoratti and V. Murino, "A real-time versatile roadway path extraction and tracking on an fpga platform," *Computer Vision and Image Understanding*, vol. 114, p. 11641179, 03 2010.

[8] L. D. Stefano, M. Marchionni, and S. Mattoccia, "A fast area-based stereo matching algorithm," *Image and Vision Computing*, vol. 22, pp. 983–1005, 03 2004.

[9] A. Aguilar-Gonzalez, M. Perez-Patricio, M. Arias-Estrada, J.-L. Camas-Anzueto, H.-R. Hernandez-de Leon, and A. Sanchez-Alegria, "An fpga correlation-edge distance approach for disparity map," in *Electronics, Communications and Computers (CONIELECOMP), 2015 International Conference on*, pp. 21–28, Feb 2015.

[10] FeiyangCheng, HongZhang, DingYuan, and MinguiSun, "Stereo matching by usingthe global edge constraint," *Neurocomputing*, vol. 131, pp. 217–226, 11 2013.

[11] H. Y. Jung, H. Park, I. K. Park, K. M. Lee, and S. U. Lee, "Stereo reconstruction using high-order likelihoods," *Computer Vision and Image Understanding*, vol. 125, pp. 223–236, 04 2014.

[12] N. ISAKOVA, S. cuk BASAK, and A. SONMEZ, "Fpga design and implementation of a real-time stereo vision system," *Circuits and Systems for Video Technology, IEEE Transactions on*, vol. 20, pp. 1–5, 07 2010.

[13] S. Jin, J. Cho, X. D. Pham, K. M. Lee, S.-K. Park, M. Kim, and J. W. Jeon, "Fpga design and implementation of a real-time stereo vision system," *Innovations in Intelligent Systems and Applications (INISTA), International Symposium on*, vol. 125, pp. 15–26, 01 2012.

[14] J. Woodfill and B. V. Herzen, "Real time stereo vision on the parts reconfigurable computer," *IEEE Symposium on Field-Programmable Custom Computing Machines*, vol. 5, pp. 201–210, 04 1997.
[15] A. Darabiha, W. MacLean, and J. Rose, "Reconfigurable hardware implementation of a phase-correlation stereo algorithm," *Journal of Machine Vision and Applications*, vol. 17, pp. 116–132, 03 2006.
[16] J. Diaz, E. Ros, F. Pelayo, E. Ortigosa, and S. Mota, "Fpga based real-time opticalflow system," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 16, pp. 274–279, 02 2006.
[17] H. Niitsuma and T. Maruyama, "High-speed computation of the optical flow," *Lecture Notes in Computer Science: Image Analysis and Processing*, vol. 3617, pp. 287–295, 09 2005.
[18] S. Lee, J. Yi, and J. Kim, "Real-time stereo vision on a reconfigurable system," *Lecture Notes in Computer Science: Embedded Computer Systems*, vol. 3553, pp. 299–307, 07 2005.
[19] D. Masrani and W. MacLean, "A real-time large disparity range stereo-system using fpgas," *IEEE International Conference on Computer Vision Systems*, pp. 13–19, 01 2006.
[20] S. S. Kumar and B. N. Chatterji, "Stereo matching algorithms based on fuzzy approach," *International Journal of Pattern Recognition and Arti cial Intelligence*, vol. 16, pp. 7:883–899, 2002.
[21] H. Ghazouani, R. Zapata, and T. Moncef, "Fuzzy sets based improvement of a stereo matching algorithm with balanced correlation window and occlusion detection," in *2010 International Conference on Image Processing, Computer Vision, & Pattern Recognition, IPCV 2010*, pp. 12–15, Jul 2010.
[22] C. Zitnick and T. Kanade, "A cooperative algorithm for stereo matching and occlusion detection," *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, vol. 22, pp. 675–684, Jul 2000.
[23] C. Georgoulas and I. Andreadis, "A real-time fuzzy hardware structure for disparity map computation," *Journal of Real-Time Image Processing*, vol. 6, no. 4, pp. 257–273, 2011.
[24] J. Olivares, J. Hormigo, J. Villalba, and I. Benavides, "Minimum sum of absolute differences implementation in a single fpga device," *Lecture Notes in Computer Science*, vol. 3203, pp. 986–990, 09 2004.
[25] D. K. Hoa, L. Dung, and N. T. Dzung, "Efficient determination of disparity map from stereo images with modified sum of absolute differences (sad) algorithm," *International Conference on Advanced Technologies for Communications*, pp. 657–660, 09 2013.
[26] S. Wong, S. Vassiliadis, and S. Cotofana, "Sad implementation in fpga hardware," *Technical Report Research Report, Delft University of Technology*, 05 2006.
[27] C. Colodro-Conde, F. J. Toledo-Moreo, R. Toledo-Moreo, J. J. Martnez-lvarez, J. G. Guerrero, and J. M. Ferrndez-Vicente, "Evaluation of stereo correspondence algorithms and their implementation on fpga," *Journal of Systems Architecture*, vol. 60, pp. 22–31, 11 2013.
[28] L. Zadeth, "Fuzzy sets," *Information Control*, vol. 8, pp. 338–353, 1965.
[29] http://vision.middlebury.edu/stereo/
[30] D. Scharstein and R. Szeliski, "A taxonomy and evaluation of dense two-frame stereo correspondence algorithms," vol. 47, pp. 7–42, apr 2002.
[31] C. Georgoulas, L. Kotoulas, G. C. Sirakoulis, I. Andreadis, and A. Gasteratos, "Real-time disparity map computation module," *Microprocessors and Microsystems*, vol. 32, pp. 159–170, 10 2008.
[32] M. Gong and Y.-H. Yang, "Near real-time reliable stereo matching using programmable graphics hardware," *Computer Vision and Pattern Recognition, IEEE Computer Society Conference on*, vol. 1, pp. 924–931, 06 2005.
[33] C. Georgoulas and I. Andreadis, "Fpga based disparity map computation with vergence control," *Microprocessors and Microsystems*, vol. 34, pp. 259–273, 06 2010.
[34] M. W. W. K. Martin Humenberger, Christian Zinner and M. Vincze, "A fast stereo matching algorithm suitable for embedded real-time systems," *Computer Vision and Image Understanding*, vol. 114, p. 11801202, 03 2010.
[35] P. C. Stefania Perri and G. Cocorullo, "Adaptive census transform: A novel hardware-oriented stereovision algorithm," *Computer Vision and Image Understanding*, vol. 117, pp. 29–41, 04 2013.
[36] C. Ttofis, S. Hadjitheophanous, A. S. Georghiades, and T. Theocharides, "Edge-directed hardware architecture for real-time disparity map computation," *IEEE TRANSACTIONS ON COMPUTERS*, vol. 62, pp. 690–704, 04 2013.
[37] S. Jin, J. Cho, X. D. Pham, K. M. Lee, S.-K. Park, M. Kim, and J. W. Jeon, "Fpga design and implementation of a real-time stereo vision system," *IEEE TRANSACTIONS ON CIRCUITS AND SYSTEMS FOR VIDEO TECHNOLOGY*, vol. 20, pp. 15–26, 2010.

**Madaín Pérez-Patricio** received the Ph.D. degree of Automation and industrial computing 2005, University of Sciences and Technologies of Lille, France. Since september 1997 he is research professor in department of postgraduate and research, Institute of Technology of Tuxtla Gutirrez, México. His primary research interest include computer vision and reconfigurable computing.

**Abiel Aguilar-González** received the B.Eng. degree in Mechatronics in June 2012, Universidad Politécnica de Chiapas, Tuxtla Gutiérrez, México. In June 2015 he received the M.Sc. degree in mechatronics engineering with highest honors, Instituto Tecnológico de Tuxtla Gutiérrez, Tuxtla Gutiérrez, México, where he worked with real-time FPGA-based system design for stereo matching algorithms. He is currently pursuing his Ph.D. degree in Computer Science at the reconfigurable computing laboratory of the Instituto Nacional de Astrofísica Óptica y Electrónica, Chulula, México. His research interests are mainly real-time image processing, real-time FPGA-based system design, machine learning and fuzzy logic applications.

**Miguel Arias-Estrada** obtained his B.Eng. in Communications and Electronics, and his M.Eng in Digital Sytems at the FIMEE (University of Guanajuato) in Salamanca, Gto. in 1990 and 1992 respectively. In 1998, he obtained his Ph.D. degree at the Computer Vision and Systems Laboratory of Universit Laval (Quebec city, Canada). He was a professor-researcher at the Computer and Systems Laboratory at Laval University where he worked on the development of a Smart Vision Camera. Since 1998 he is with the Computer Science department of INAOE (National Institute of Astrophysics, Optics and Electronics, Puebla, Mexico) where he continues his research on FPGA architectures for computer vision. His interests are Computer Vision, FPGA and GPU algorithm acceleration for 3D and machine vision.

**Héctor Ricardo Hernández De León** is Research Professor at Department of Electrical & Electronics Engineering of Institute of Technology Tuxtla Gutierrez, Chiapas, Mexico (ITTG). He is Electronics Engineer by the National Polytechnic Institute of Mexico City (IPN). He obtained an MSc and PhD in Automated Systems at the National Institute of Applied Sciences INSA-Toulouse-France. He is integrated into the research group "Process Automation" of MSc program in Mechatronics Engineering at ITTG. Interest fields are supervisory control, diagnosis of complex processes. His specialty is the automatic control applied to industrial processes. Some of current works are: Intelligent control and diagnosis of potable water plants and Biodiesel diagnosis and prognosis.

**Jorge Luis Camas Anzueto** received the PhD degree from Instituto Nacional de Astrofísica Óptica y Elctrónica (INAOE), Puebla, México in 2004. He is currently a researcher in Maestría en Ciencias en Ingeniería Mecatrónica (MCIM) of the Instituto Tecnológico de Tuxtla Gutiérez, Chiapas, México. His research interests include optical sensors, fiber sensors, and optoelectronics.

**Juan Antonio de Jesús Osuna-Coutiño** received the B.Eng. degree in Industrial Productivity in June 2011, Instituto Tecnológico de Tuxtla Gutiérrez, Tuxtla Gutiérrez, México. In June 2015 he received the M.Sc. degree in mechatronics engineering with highest honors, Instituto Tecnológico de Tuxtla Gutiérrez, Tuxtla Gutiérrez, México. His research interests are GPU algorithm acceleration, shortest path problem and fuzzy logic applications.