Journal of Theoretical and Applied Information Technology

15th October 2023. Vol.101. No 19
© 2023 Little Lion Scientific

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

EMBEDDED FPGA HARDWARE IMPLEMENTATION OF A PREPROCESSING ALGORITHM FOR SURVEILLANCE IMAGES

ISSAM BOUGANSSA[1], HICHAM BENRADI[2], ADIL SALBI[3], MOHAMED SBIHI[4], ABDELALI LASFAR[5]

[1,2,3,4,5] Laboratory LASTIMI, High School of Technology SALE Mohammed V University RABAT, MOROCCO

Email: [1]Issam.bouganssa@gmail.com, [2]hicham_benradi@um5.ac.ma [3]a.salbi89@gmail.com, [4]Mohamed.sbihi@yahoo.fr [5]abdelali.lasfar@est.um5.ac.ma

ABSTRACT

Surveillance images have become essential for deciphering many accidents and offenses, however, the major obstacle in this area is their quality, which is often obtained via low-resolution surveillance cameras, or in difficult light conditions. This work aims to implement in real-time on an FPGA-type embedded system, pre-processing algorithms to improve the quality of surveillance images by normalizing and enriching histograms, followed by edge detection algorithms, using software and hardware tools. The objective of this improvement is to allow professionals to dissect the necessary information more easily. To do this, it was decided to program the algorithms with a VHDL-type hardware description language and test them on more efficient and faster tools called Xilinx System Generator (XSG) which allows several algorithms to be tested on software, before implementation. This implementation is done on a XILINX SPARTAN-6 FPGA board using the ISE Design Suite tool from Xilinx.

Keywords: *Pre-Processing, Normalization, FPGA, Edge-Detection, VHDL.*

1. INTRODUCTION

Currently, the image is one of the most important means used by man to communicate with others, because it provides valuable information on the scenes or objects filmed. In the field of crime, in recent years several public places have recognized the installation of surveillance cameras, thanks to these installations the images have become essential elements for the establishment of a good investigation for crimes and accidents. Image processing refers to methods and techniques operating on the image to extract more relevant information, improve the visual aspect of the image, or the coding and decoding of images for storage or transmission [1]. Different image processing applications are used such as segmentation or edge detection and texture analysis for the identification of objects in the image or facial recognition. The detection of vehicle registration numbers, and also image compression both reduce storage costs and network transfer times.

The work presented here is part of the general context of improving the quality of surveillance images, by improving the contrast with pre-processing algorithms based on histogram normalization and enhancement, to detect the edges of objects correctly using an embedded system. The objective of this work is the hardware implementation in an embedded system represented by a reprogrammable circuit of the FPGA type of the Xilinx family, using a combination of hardware and software components. The implemented algorithms are of temporal type, for the pre-processing part they are based on the variation of the intensity by the calculation of histogram and the normalization of the contrast, and for the processing part the algorithms are based on the detection of the contours by application of convolution masks, these methods are followed by judicious thresholding allowing the contours to be isolated from the rest of the image [1,2].The resulting images after quality enhancement are displayed in real-time on a VGA monitor. The implemented algorithms provide the performance needed for processing image sequences in real-time while maintaining the flexibility of the system to support adaptive algorithms. These performances are confirmed by the results of real-time edge detection applied to images in the surveillance domain as well as to other images in different domains.

2. THE DIGITAL IMAGE PREPROCESSING SYSTEM

A digital image processing system is composed of several steps, starting with acquisition until display (figure 1):

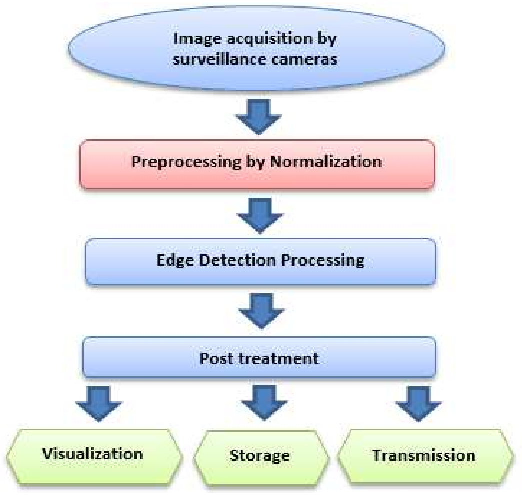


Image acquisition by surveillance cameras
Preprocessing by Normalization
Edge Detection Processing
Post treatment
Visualization
Storage
Transmission

Figure. 1. Digital Image Processing System

Image acquisition is the first phase of our image processing system, to be able to manipulate an image on a computer system, it is above all necessary to subject it to a transformation that will make it readable and manipulable by this system. This operation is done by the various image acquisition devices and surveillance cameras for our acquisition case [3].

Pre-processing is the second step used in this image processing system, gathers all the techniques aimed at improving the quality of an image, and so is the preparation of the images before processing (edge detection for this case), Therefore, the starting data is the initial image and the result is also an image with higher quality. The pre-processing methods are very numerous, they concern:

- Restoration: This consists of recreating the modified image by eliminating defects due to a source. Its goal is to obtain an image that is as close as possible to the ideal image that would have been obtained if the acquisition system was perfect.
- Filtering: This consists of removing the noise present in an image by studying, for each pixel, the intensity values in its vicinity.
- Contrast enhancement: consists of modifying the visual characteristics of poor quality images (poor contrast): images that are too dark or too bright, in order to satisfy and facilitate their interpretation by the human eye.

3. PREPROCESSING METHODS BY HISTOGRAM ENHANCEMENT AND NORMALIZATION

The pre-processing methods are very numerous but the most used in poor-quality surveillance images are those of contrast enhancement, this contribution is therefore interested in contrast enhancement methods. Since the interpretation of the (dynamic) contrast is like a spread of the histogram for the different pixels that constitute the image to be processed.

The idea of pre-processing, a bad-quality image, consists in working on the histogram, to define a function that allows calculating a new value of each pixel, allowing to highlight the details present in an image, Histograms are frequently used to perform this type of operation.

3.1 Histogram Principle For A Digital Image

The histogram is a statistical graph for representing pixels according to their light intensity value, i.e. the number of pixels for each light intensity. The histogram of the levels of gray of an image is a function that gives the frequency of appearance of each level of gray in the image, this histogram will be represented by a graph having 256 values in abscissa, and the number of pixels of the image in ordinates. The more pixels to the left, the darker the image. The more pixels to the right, the clearer the image [3,4].

For an RGB-coded color image, several histograms are necessary, one histogram representing the distribution of the luminance, and three histograms representing the distribution of the values, respectively of the red, blue, and green components. The more pixels there are to the left, the darker the image and the less striking the color. The more pixels to the right, the clearer the image and the denser the color. The Histogram is a very important tool for the pre-processing of images because its modification allows adjusting the dynamic contrast of grayscale or colors in an image, it can be used to improve the quality of an image (Image Enhancement), usually to have good visibility and clarity.

3.2 Methods For Improving Histograms

To modify the characteristics of the image (boost the contrasts in general), a general approach consists in applying a function that associates with each intensity value in the image a new value. This function will modify the histogram of the image to be processed.

The idea is to improve the contrast of the image, when the conditions for acquiring an image are poorly controlled, i.e. the image is too dark or too bright. The quality of the image can be improved by modifying the histogram (modifying the distribution of levels), by calculating for each pixel a new value that depends only on the starting value of the pixel, by several methods. As part of this research, some examples of image enhancement by histogram modification are represented here, these examples are focused on the normalization method.

3.3 Histogram Normalization And Image Enhancement

Normalization is a method that consists of spreading the intensities (dynamics) of an image, included in the interval [a,b] over the 256 available gray levels. This is called histogram stretching, which improves the contrast. This transformation in the histogram makes it possible to discern details not visible on the original image by using the formula to stretch the histogram.

$$I'(i,j) = a + b \times \frac{I(i,j) - \min_{ij} I}{\max_{ij} I - \min_{ij} I} \qquad (1)$$

I'= histogram of the stretched image
a = the lowest intensity value
b = the strongest intensity value
I = histogram of the original image
Min = the value of the min intensity histogram
Max = the max intensity histogram value

Histogram stretching has almost no effect on images whose histogram occupies almost the entire grayscale range. Below is an example of the application of this function figure 2.

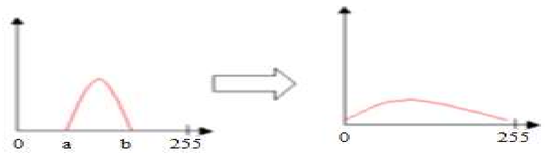


Figure. 2. The Histogram Stretching Process

4. IMAGE PROCESSING BY EDGE DETECTION AFTER ENHANCEMENT

4.1 Methods Based On Convolution Products For Edge Detection

The contours also have their own particularities, which can be used to forecast which detector will be most successful in addition to the visual features of color and texture [5]. To determine how effectively the detectors are working, some errors encountered during the detection of contours are mentioned below (Perfect edge Fig 3A):

- Exclusion of some pixels from the contour that needs to be detected. The number of missed pixels in relation to the overall number of pixels in the ideal contour is counted to get the value (Fig. 3B).
- Location: This mistake happens when a pixel on an unambiguous ideal contour is off-center. The entire distance between the detected and ideal contours is counted to determine its size. (Fig 3C).
- Multiple reactions by multiple contours being detected. It is determined by calculating the proportion of ambiguous pixels to unambiguous pixels (Fig. 3D).
- Sensitivity: This mistake corresponds to spurious contours found near the ideal contour and is frequently related to noise. The number of incorrect pixels and the global number of detected contours are counted to determine it (Fig. 3E).

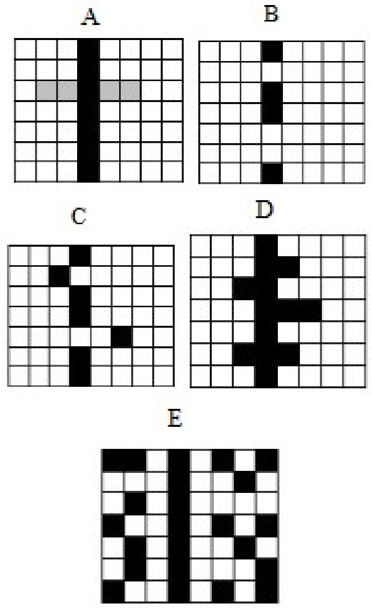


Figure. 3. Examples of Edge mistakes; A: Ideal Edge, B: missed pixels, C: off-center pixels, D: numerous replies, and spurious Edge in E.

4.2 Edge Detection Principle Based On The Calculation Of Convolution Products

In the realm of computational imaging, a local intensity variation is the main source of data. The pixel's [i, j] gradient vector mathematical function [6] is used to measure it. In an orthogonal system of coordinates (Oxy), where (Ox) is the horizontal axis and (Oy) is the vertical one, the picture gradients (or rather the brightness f) from any position or pixel coordinates (x, y) [5–6] are given below by expression 2:

$$Grad\ f = \nabla f = \begin{pmatrix} \frac{\partial f}{\partial x} \\ \frac{\partial f}{\partial y} \end{pmatrix} \quad (2)$$

The gradient module quantifies the importance of the Edges presented pixels, i.e. the amplitude of the intensity jump observed in different pixels of the image:

$$\|\nabla f\| = \sqrt{(\frac{\partial f^2}{\partial x}) + (\frac{\partial f^2}{\partial y})} \quad (3)$$

The contour visible within the pixels of the picture is determined by the gradient's α_o orientation. In fact, the gradient's direction is always opposite to the direction of the contour:

$$\alpha_o = \arctan(\frac{\partial f/\partial y}{\partial f/\partial x}) \quad (4)$$

After pre-processing by normalization, the principle of edge detection using the gradient is to calculate, first, the gradient of the image in two orthogonal directions, then the gradient modulus. Similarly, the next step consists in making a selection of the most marked contours, that is to say, the points of strongest contrast with an adequate thresholding picture below:

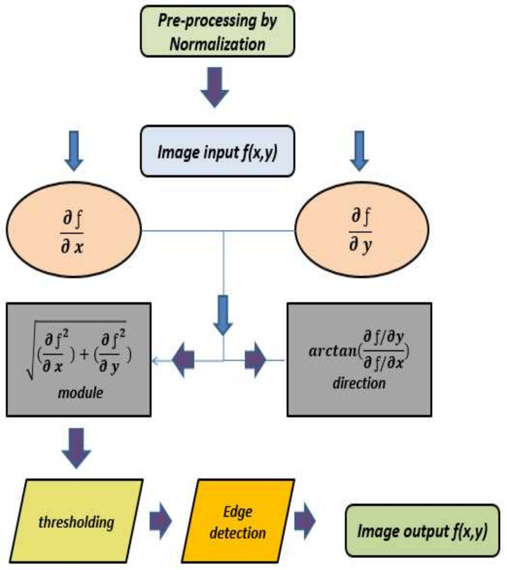

Pre-processing by Normalization
Image input f(x,y)
$\frac{\partial f}{\partial x}$
$\frac{\partial f}{\partial y}$
$\sqrt{(\frac{\partial f^2}{\partial x})+(\frac{\partial f^2}{\partial y})}$
module
$arctan(\frac{\partial f/\partial y}{\partial f/\partial x})$
direction
thresholding
Edge detection
Image output f(x,y)

Figure. 4. Principle of edge detection.

4.3 Step Of Thresholding

The picture's origin was determined using the earlier algorithmic techniques, which makes it possible to highlight these edges well. At this level, the resulting image is expressed in gray-levels, indicating here the importance of each luminance intensity [7].

To isolate the edges from the rest of the image, a new step is necessary to obtain more precise information allowing to test the presence of the edges. The resulting image IB (i,j) of this processing is in black and white. The white pixels (value 1) indicate the presence of an outline and the absence of black pixels (value 0).

Converting a grayscale image into a black and white binarization image after the IM (i,j) image generated by convolution products, requires setting a thresholding parameter and determining a threshold value to represent the most significant contours and avoid false contours.

The value here designated S, is selected to show the Edges that represent the combined histogram of the gray-level imagery found to be the most important. The resultant pixel value is 1 if the picture's pixel level surpasses the chosen threshold. If not, the value of the pixels is set to 0:

$$\begin{aligned} \boldsymbol{I}_B(i,j) &= 1 \quad \text{if } \boldsymbol{I}_M(i,j) \geq S \\ &= 0 \quad \text{else} \end{aligned} \quad (5)$$

5. IMAGE ENHANCEMENT AND SIMULATION OF EDGE DETECTION OPERATORS BEFORE AND AFTER PRE-PROCESSING

In this part, the application of histogram normalization is presented for contrast enhancement of multiple images, followed by edge detection masks on the same surveillance images with and without pre-processing, the pre-processing is performed by the implementation of these normalization algorithms. The objective of this application is to quantify the importance of normalization in the image processing process.

In the figures below, the landscape image chosen represents several people in a means of transport before and after pre-processing, in figure 5 to 8 a portrait image is illustrated of a single person before and after pre-processing, and the last image to be used is that of the famous Lena (standard image widely used in image processing) before and after pre-processing.

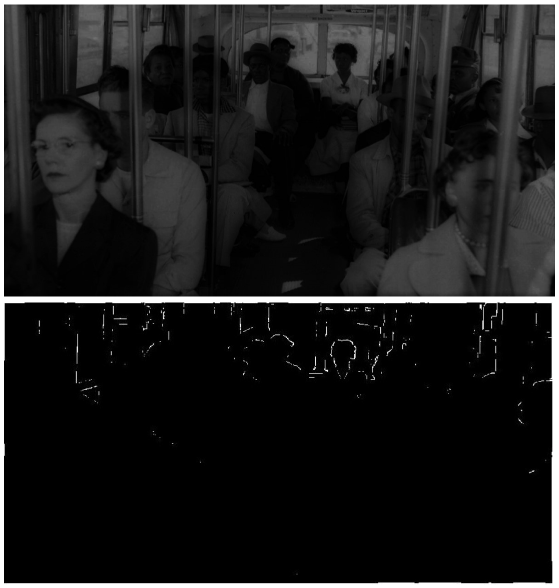


Figure. 5. Example Of Landscape Image With Edge Detection Before Preprocessing

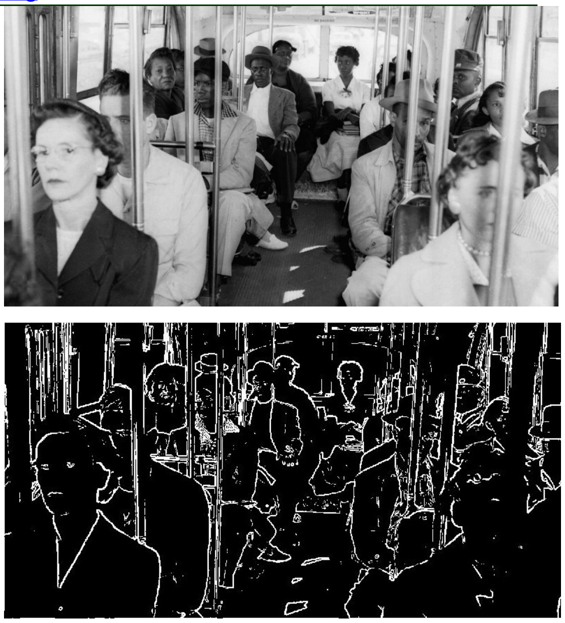
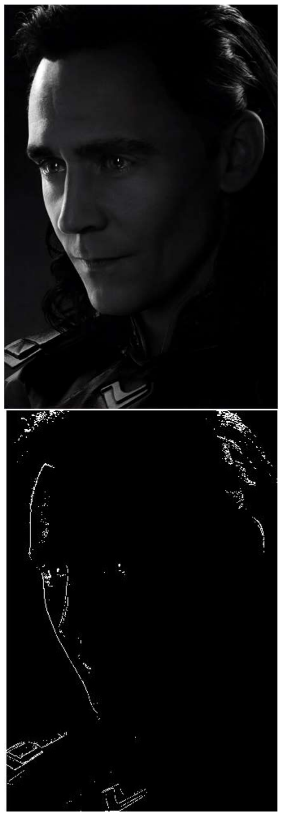


Figure. 6. Example Of Landscape Image With Edge Detection Before After Preprocessing



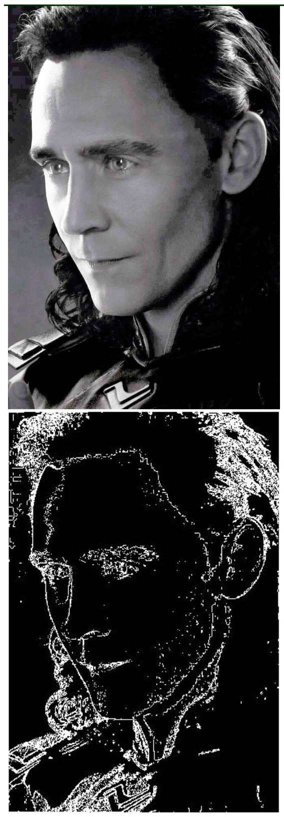
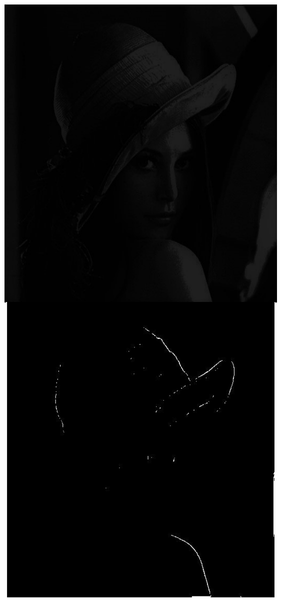


Figure. 7. Example Of Portrait Image With Edge Detection Before And After Preprocessing



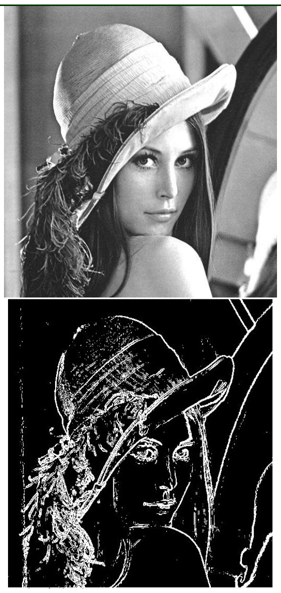


Figure. 8. Image Of Famous Lena With Edge Detection Before And After Preprocessing

These results showed that the histogram normalization is a very important tool in image processing because its modification allows adjusting the dynamic contrast of gray-levels or colors in an image, it was used to improve the quality of his images and retouching in general in order to have good visibility.

Similarly, vehicle registration detection and fingerprint recognition can be used to help solve many crimes and accidents. The highlighting of marks that indicate an object's outlines in an image acquired by surveillance cameras could be employed as a diagnostic tool, to distinguish image regions, to extract compressed information that is frequently relevant for analysing the image, and more, the texture analysis for the identification of objects in the image or facial recognition.

6. HARDWARE AND SOFTWARE REQUIREMENTS FOR THE PRE-PROCESSING AND PROCESSING OF SURVEILLANCE IMAGES

6.1 Hardware Solution For The Implementation Of An FPGA Architecture

To meet the real-time needs of a vast array of audio-visual and image-processing applications on embedded systems, there have been suggested hardware implementation strategies for reconfiguration architectures. The majority of currently available solutions rely on processor-based computational units with 32-bit inputs and outputs. These approaches, nevertheless, do not offer the data's memory management process or are intended to be achieved by a software component [8].

Considerable processing power is needed for real-time image processing on embedded systems. For instance, the common image.jpg format for surveillance cameras has a resolution of 30 photos per second and a resolution of 640*480, or around 0.30 megapixels, every image. Larger pictures are possible [9], and the amount of processing needed per pixel depends on the pre-processing and processing algorithm used to estimate the computation time.

Processing demands will rise when pictures with excellent resolution are used more frequently. Standard photographs with high resolution often have ten times more pixels each frame. There is around a ten-fold increase in computing load. They need multiple DSP processors or extremely expensive high-end DSP processors to process them. FPGAs offer an alternate real-time image processing solution in this case. For the effective implementation of pre-processing and image processing techniques, the FPGA effectively enables large volumes of information to flow for simultaneous processing.

6.2 Software Solution For Implementation

A hardware description language (HDL) is a type of computer language, definition vocabulary, or model vocabulary used in semiconductors for formalizing the structure and layout of digital logic systems in general.

An HDL program allows for a virtual representation of a circuit's function in addition to describing the circuit's organization, design, and function. Indeed, HDLs are standardized textual forms for expressing the spatial and temporal organization and behaviour of electronic systems [9]. An HDL's grammar and semantics, like those of all coding languages, offer explicit notations for expressing concurrency. However, since time is the primary characteristic of a device's layout and is not implicit in most other languages, an HDL is required.

For the creation of complicated reasoning devices, the VHDL language was developed. Its certification by the IEEE under the references (IEEE 1076.87 and 1164.93), It makes it a unique language for documentation, syntheses, demonstrating, and simulation, is largely responsible for its popularity. Because VHDL is written independently of any specific technology or design process, it makes it easier to move between the various useable technologies, which are always advancing [10, 11]. The Xilinx ISE software can generate one of the three types of templates for the VHDL description:

- The operational framework, which makes no mention to any implementation, defines how an object works using a sequential algorithm or a truth table with simulator.
- The information flow framework uses simple logic formulas to represent the flow between input and output at the bit level.
- The structural model of an object closely resembles the diagram by representing the object's composition as a series of interconnected fundamental blocks.

The most effective and quick tool is called Xilinx System Generator (XSG), and it may be used to facilitate FPGA development for different pre-processing of images and computation techniques, which allows testing of multiple algorithms on software (processing of images) before the Xilinx hardware implementation.

7. XILINX ARCHITECTURE FPGA HARDWARE REALIZATION.

The Xilinx Nexys-3 architecture was employed for the suggested technique's realization. Depending on the Xilinx FPGA Spartan-6 LX16 (Fig. below) and using a VGA display to show the outcomes, the Nexys-3 is a full, prepared-to-use digital device creation environment [11].

In comparison to the Nexys2 Spartan-3 500E FPGA employed in earlier studies, Nexys-3 integrated Spartan6 delivers more than 50% larger capacity, more performance, and more resources. It has been tailored for exceptionally well logic. Capabilities of the Spartan-6 LX16 comprise:

- 2,278 slicing with eight 6-input LUTs with eight switches all of them
- 576 KB of speedy blocking RAM, two clock tiles (four DCM and two PLL),
- 32 DSP segments,

- A clock rate of 500 MHz or higher. Illustration of the Spartan-6 device at 100 MHz.

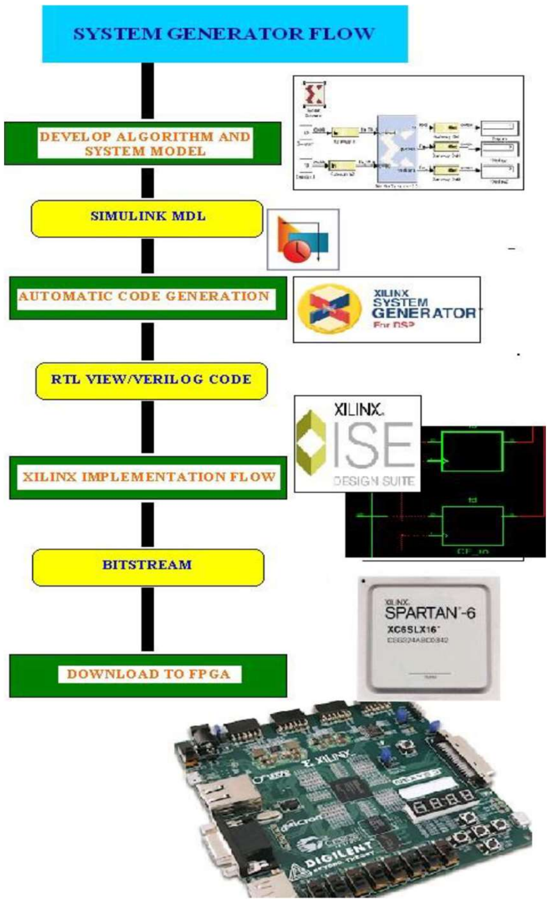

SYSTEM GENERATOR FLOW
DEVELOP ALGORITHM AND SYSTEM MODEL
SIMULINK MDL
AUTOMATIC CODE GENERATION
XILINX SYSTEM GENERATOR For DSP
RTL VIEW/VERILOG CODE
XILINX ISE DESIGN SUITE
XILINX IMPLEMENTATION FLOW
BITSTREAM
XILINX SPARTAN-6 XC6SLX16
DOWNLOAD TO FPGA
DIGILENT

Figure. 9. Nexys-3 device Architecture [11]

The Nexys-3 card includes an increased range of accessories in addition to the Spartan-6 FPGA device, such as Micron's 32-bit non-volatile phase-change memory, 10/100 Ethernet, 16MB Cellular RAM, USB UART, a USB host connection for mouse and keys, and an upgraded high-speed expanding slot. The methods were created using the Xilinx ISE interface, and all of the code modules are written in VHDL including comprehensive drawings for every component:

7.1 "Memory" And "Memory Playback" Program Block

The "Memory" block that exists in the IP (intellectual property) a portion of the Xilinx software is set up based on the dimension of the image that will be saved after being collected from a surveillance system. Every cell in memory has 8 bits, of which 8 are used for encoding the value of every RGB pixel.

Every of the four DDR2-Micron information libraries in Xilinx platforms is structured into a unit array (2^13 rows × 2^10 columns), with each cell consisting of 64 bits, and every bank is accessed by one of the two address lines (BA0, BA1). It is necessary to utilize a multiplexer at the DDR2 controller level because the lines A0-A12 address the data rows and A0-A9 reference the columns [11].

Every DDR2 storage cell may hold 8 pixels if the luminance signal (on 8 bits) is the only thing you modify. Similar to this, 8 × 2 ^ 10 pixels can fit on a storage line, and an image of 256 256 pixels takes up 8 storage lines [11–13]. We can see given this design that DDR2 prefers analysing information messages, even multiple messages at once. Using a FIFO connection to an interaction-based graphical interface, the logical user connects to the storage driver.

Three related channels make up each connection:

- This bus, which is a FIFO control/address bus, receives read/write instructions along with the associated storage location.
- A writing information bus of the FIFO variety that takes write data in response to a writing instruction given on the control bus.
- A FIFO-type read bus, on which the data is read to the working stages using a Read Enabled instruction.

Because the picture does not always have to be identical in size as the display, the "Memory Playback" program block is helpful for ensuring that the picture is positioned correctly on the display.

7.2 Program Block "Synchronization" And "VGA Screen"

The software lets you sync both vertical and horizontal pixel scans on the screen. The timing of the rate of slewing is controlled by signals. The scanned time for a line is determined by the horizontal synchronization signal, whereas the scan time for the whole display is determined by the vertical sync information. Images are created on the monitor screen by modifying these impulses [12, 13].

Similar to this, five 10-bit coded signals red (3 bits), green (3 bits), blue (2 bits), horizontal synchronization, and vertical sync—are used by the application to define the VGA display. The RGB signal, which is made up of the three color information, is used to control the color of a pixel on the screen. Each color digitally needs to be given a voltage between 0.7 and 1.0 volts in order to change the color concentrations and create different colors.

7.3 Program Block Normalization Preprocessing Algorithm

It is possible to improve the perception of certain images by simply adding an adjustment to the lighting level. The multi-dimensional R|G|B image signals for color pictures resolve similarly to how the one dimension rather image information for shades of gray resolves for shades of gray. Mathematically, this is expressed as explained in the figure below:

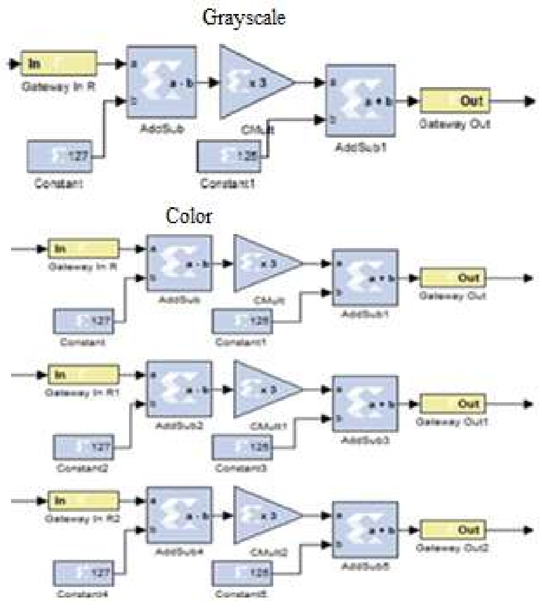


Figure. 10. Grayscale and Color Contrast Stretching Algorithm

The range of brightness values that are present changes in response to changes in contrast in a picture. This has the same effect as widening or narrowing the histogram's range around the middle value. To create color contrast photos, the same method can be used. In Figure 10, it is depicted. For color images, the stretching of contrast enlarges the contrast, hue, and saturation values of each pixel for each R, G, and B information.

7.4 Program Block Processing Systems

The contour detection technique that was introduced in part 4 is applied in this section of the program. Always, the converted pixels are fewer than the original pixels. The system is clocked 1/3 more quickly than usual in the event of the 3X3 filter creation. This block contains the internal delay, which causes some lines at the bottom of the image to be moved upward. The image is translated down for the same amount of rows to prevent this problem. After filtering, the full image is recovered using this.

The upper bound ensures that FIFO list-dependent actions can be carried out within a frame's real-time processing time (see image under).

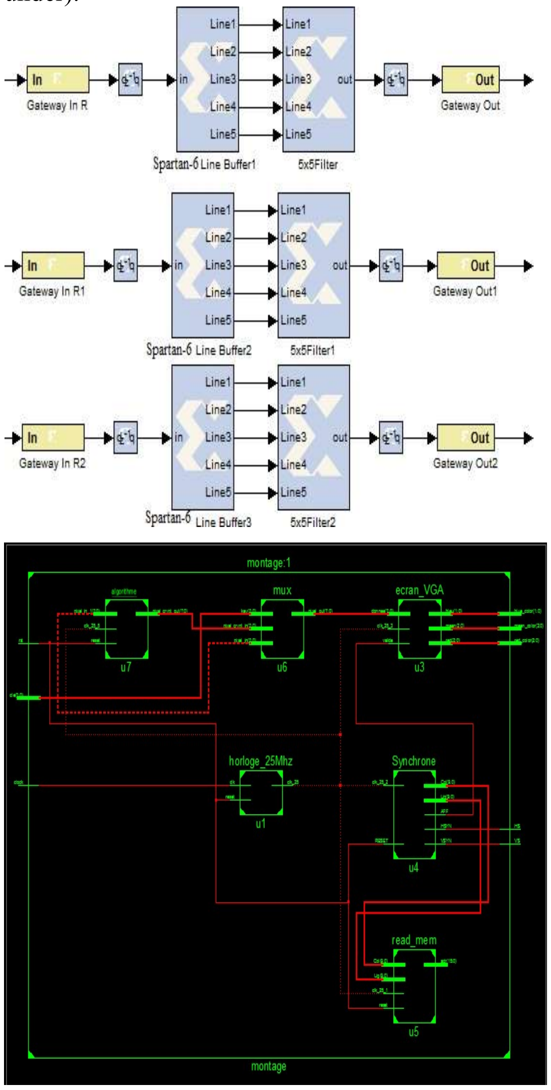


Figure. 11. Edge detection algorithm for color images and blocks implemented on Xilinx ISE

Multiple built-in M4K memory blocks that are each 4 kbps in size are used for implementing FIFOs. The Spartan-6 device's

RAM chips are where the image of the edge detection result is kept [14]. When two bits are allotted for every pixel position, it is possible to temporarily store the edge image in these substantial internal RAM units.

Figure 11 depicts a schematic of the many VHDL-programmed blocks that the Xilinx ISE software created [11–14]. Two forms of prototype can be implemented using FPGA devices. The first type exclusively employs IP computers, whereas the second variety employs an interface like Micro Blaze that enables the reusing of pre-programmed subroutines.

Table 1 presents some statistics generated by the ISE-XILINX software during the implementation of the proposed programs on the Spartan-6 FPGA.

Table 1. Device Utilization Summary

Slice logic utilization	Used	Available	Utilization
Cells of slice Registers	71	18224	01%
Cells of slice LUTs	99	9112	01%
Cells used as Store	00	2176	0%
Cells Used as logic	98	9112	01%
Cells of occupied slices	36	2278	01%
Cells MUX cys used	43	4556	01%
Cells with an unused Flip-Flop	39	98	39%
Cells with an unused LUT	05	98	05%
Cells of fully used LUT-FF pairs	66	98	67%
Cells of slice register sites lost to control set restrictions	34	18224	01%
Cells of bonded IOBs	27	232	11%
Cells of Loced IOBs	15	20	75%
Cells of SRAM B16BWERs	26	32	81%
Cells of SRAM B8BWERs	00	64	00%
Cells of BUFIO2/ BUFIO2-2CLKs	00	32	00%
Cells of BUFIO2FB/ BUFIO2FB-2CLKs	00	32	00%
Number of BUFG/ BUFG-MUXs	04	16	25%

The number of processors that can be combined on a single component depends upon its available resources. The flip-flop pair for this architecture is analogous to a slice's LUT and flip-flop pair. For a recording item, an instruction set is a specific pair of the clock signals (reset and enable).

8. THE RESULTS OF THE DEBATE AND ITS EXECUTION

To demonstrate the edited photos, before and after the implementation of the normalization and enhancement pre-processing algorithms, A Xilinx Spartan-6 FPGA circuit is employed and is directly coupled to a VGA display, the latter is powered by a 5 volt DC adapter. Figure 12 shows the image before and after pre-processing. An image is used, generated via a surveillance camera has a 200x200 size. Each symbol cycle of the clock a pixel is determined in the fully pipelined function. More than 350 pictures per second at a size of 200x200 can be processed with this bit rate and clock frequency. Additionally, the entire chain was created using the proper technique, Algorithm Architecture.

These capabilities make an excellent candidate for a real-time hardware implementation for embedded platforms due to the regularity of the calculations, the decreased memory requirements, and the high inherent parallelism [14,15]. FPGAs are typically the least expensive embedded platforms in terms of performance and power consumption optimization, and they can contend because they are widely available, reprogrammable, and made using the most up-to-date manufacturing techniques. When the elements (money, time) needed for creation are not available for ASICs, they present an intriguing alternative. They are also competitive with sequential microprocessors due to their capacity to parallelize operations and carry out configurable functionalities [15].

The results obtained show a great improvement in the contours of the processed images, and prove the importance of the pre-processing step, especially for images acquired under abnormal conditions, in our case the images from surveillance cameras. With keeping the clock rate at 100 MHz, this system is scalable to handle high-resolution images [16]. As well as high-speed surveillance image acquisition equipment, which requires more frames per secondly, our approach can process a regular image tram up to 30 frames per second. By adding more pipeline stages at the penalty of higher resource consumption, the frequency can be raised exceeding 100 MHz [15,16].

Several extra memory are available on the Xilinx Nexys-3 platform that can be set to augment the capacity of storage on the Spartan-6 circuit: a 128 Mbit cellular RAM (pseudo-static DRAM), a 128 Mbit non-volatile parallel PCM (memory of change phase), and a 128 Mbit PC serial card.

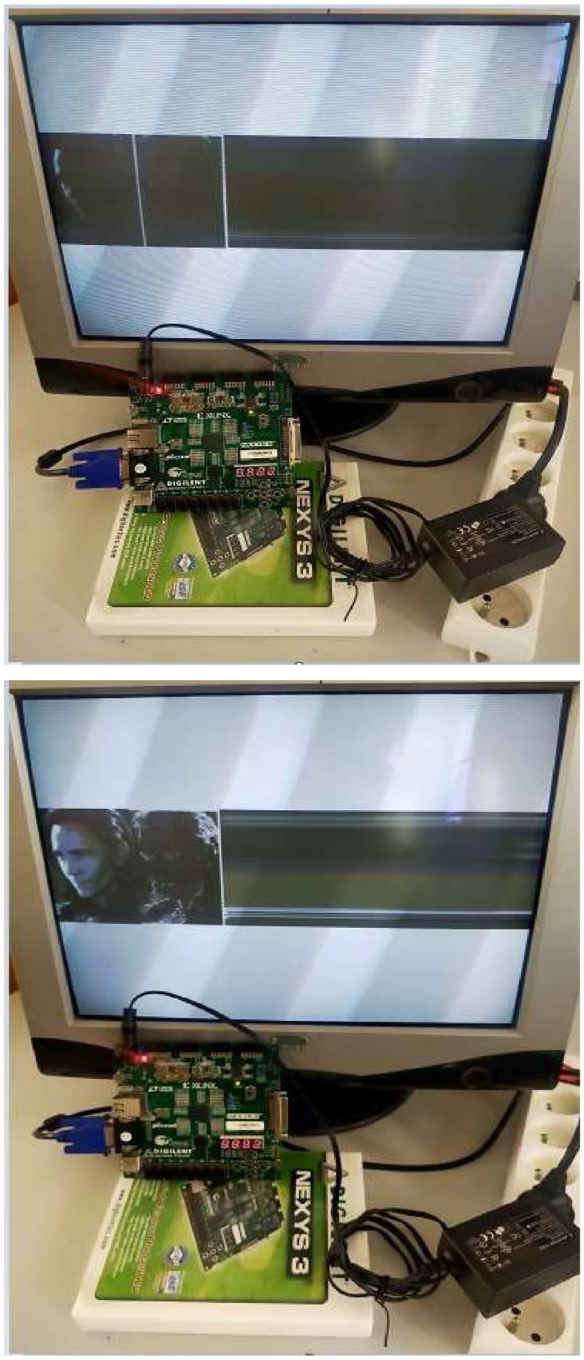


Figure. 12. The Results Of The Hardware/Software Implementation Before And After The Treatment

Indeed, to apply the improvement in processing speed and access external memories, whatever their type, a memory controller is used which acts as an intermediary between the user program and the physical memory. This controller is also used to perform command operations; calculations of physical addresses from logical addresses, recording of data and addresses in FIFO for reading/writing, and automatic refreshing.

9. CONCLUSION

This work presents a general context on improving the quality of surveillance images by contrast enhancement, this improvement and based on pre-processing algorithms by histogram normalization and enhancement in order to detect the edges of objects. In a correct manner using software and hardware tools.

The objective validated in this work is the hardware implementation in an embedded system represented by a reprogrammable circuit of the FPGA type of the Xilinx family, using a combination of hardware and software components [16]. The implemented algorithms are of temporal type, for the pre-processing part they are based on the histogram calculation and the normalization of the contrast, and for the processing part the algorithms used, are based on the detection of the contours by application of convolution masks.

The results obtained show the clear presence of all the contours that exist in the images and even in the case of very dark images. These results can be used as a diagnostic tool to characterize areas of the image and to extract information from them, reduced due to lack of lighting, often relevant to identify objects captured by surveillance cameras. When processing low-quality security photographs (30 pictures per second) at the same frequency of the clock (100 MHz), the Spartan-6 FPGA shows its capacity to handle high-resolution VGA pictures, as well as applications (artificial intelligence) which are based on computer vision.

REFERENCES

[1]. R.C. Gonzalez, R.E. Woods, "Digital Image Processing".New Jersey: Prentice-Hall 2008 v.

[2]. D. Marimont, Y. Rubner, "A probabilistic Frameword for Edge Detection and

selection of Scale", Proceedings of the International Conference of the IEEE Computer Vision, 207-214, January 1998.

[3]. J.F. Canny, "A computation approach to edge detection," IEEE Transactions on Pattern Analysis and Machine Intelligence, vol. 8, 6pp. 769-798, November 1986.

[4]. W. He and K. Yuan, "An improved Canny Edge Detector and its Realization on FPGA," Proc. of 7ili World Conference on Intelligent Control and Automation, 2008.

[5]. Boganssa, M. Sbihi and M. Zaim "Implementation on a FPGA of Edge Detection Algorithm in Medical Image and Tumors Characterization" IEEE. 2016 DOI: /10.1109/icmcs.2016.7905655, 2016.

[6]. Y. Bao, K. Song, J. Wang, L. Huang, H. Dong, Y. Yan, Journal. Visual. Com and Image. Rep 11, 2021.

[7]. Boganssa, M. Sbihi and M. Zaim "Implementation in an FPGA circuit of Edge detection algorithm based on the Discrete Wavelet Transforms" Journal of Physics: Conf. Series Vol. 870 No 012016 (2017)

[8]. Salbi and S. Bri, « VHDL Miniaturization of an Electric Rehabilitator with Dynamic Control », International Journal of Current Research, Vol. 8, Issue 08, pp.37060-37067, August, 2016.

[9]. http://www.xilinx.com/products/design-tools/ise-design-suite.html

[10]. Xie, J., Hu, K., Guo, Y., Zhu, Q., Yu, J.: On loss functions and cnns for improved bioacoustic signal classification. Ecol. Inf. 64, 101331, 2021.

[11]. Xilinx Inc., 'Xilinx Memory Interface Generator (MIG) User Guide, DDR SDRAM, DDRII SRAM, DDR2 SDRAM and RLDRAM II Interfaces', Document: UG086 (v2.0), 2007.

[12]. Ghosh, D. Chakraborty, A. Law, S. Nepal, Q. Z. Sheng, Artificial Intelligence in Internet of Thingsthe , CCAI. Transactions. Intelligence. Tech, 2018.

[13]. M. Qasaimeh, K. Denolf, J. Lo, K. Vissers, J. Zambreno, and P. H. Jones, Comparing Energy Efficiency of CPU, GPU and FPGA Implementations for Vision Kernels, in 2019 IEEE international conference on embedded software and systems, ICESS, 2019.

[14]. M. Merenda, C. Porcaco, D. Iero, Edge Machine Learning for AI-Enabled IoT Devices, Sensors. 20, 24-28, (2020).

[15]. J. Seojin, L. Wei, P. Sangun, C. Yongbeom, Automatic RTL Generation Tool of FPGAs for DNNs, Electronics. 11, 402, 2022.

[16]. Boganssa, M. Sbihi, and M. Zaim, Laplacian edge detection algorithm for road signal images and FPGA implementation, Int. J. Mach. Learn Comput, 2019.