Object Detection and Segmentation Algorithm Implemented on a Reconfigurable Embedded Platform Based FPGA

SLAMI SAADI[1] HAMZA MEKKI[2] ABDERREZAK GUESSOUM[2]

[1]Nuclear Research Centre of Birine (CRNB), Bp180, AinOussera, 17200, ALGERIA

[2]University Saad Dahleb, Signal Processing and Imaging Laboratory (LATSI), B.P.270 Blida 09000 ALGERIA

saadisdz@yahoo.fr

Abstract: - In this article, we present a mixed software/hardware Implementation on a Xilinx's Microblaze Soft core based FPGA platform. The reconfigurable embedded platform designed to support an important algorithm in image processing which is region color image segmentation for detecting objects based on RGB to HSL transformation. The proposed work is implemented and compiled on the embedded development kit EDK6.3i and the synthesis software ISE6.3i available with Xilinx Virtex-II FPGA using C++ language. The basic motivation of our application to radio isotopic images and neutron tomography is to assist physicians in diagnostics by taking out regions of interest. The system is designed to be integrated as an extension to the nuclear imaging system implemented around our nuclear research reactor. The proposed design can significantly accelerate the algorithm and the possible reconfiguration can be exploited to reach a higher performance in the future, and can be used for many image processing applications.

Key-Words: - Color images, Segmentation, Reconfigurable, Embedded, FPGA.

1 Introduction

Segmentation is with no doubt one of the largest research areas in image processing. New algorithms, algorithmic techniques, methodologies and improvements are continuously being proposed to segment images into their constituent. Color image segmentation is highly useful in many applications including image enhancement, target recognition, image indexing for content-based retrieval.

Image segmentation is often described as the process that subdivides an image into its constituent parts and extracts those parts of interest (objects) [1].

Digital radiological image is a digital image acquired by a certain radiological procedure which can be X-rays, neutron radiography, gamma camera image or a nuclear magnetic resonance image. It is a two-dimensional MxN array of non-negative integers (gray levels). For tomography, the gray level value represents the relative linear attenuation coefficient of the object.

The performance of nuclear image segmentation depends not only on the method used but also the strength of the algorithm to extract a difficult region of interest. Higher performance show finer structural or functional information of body organs and support more reliable diagnostic outcomes [8].

Our flexible implementation helps a lot for image segmentation and analysis. The design utilizes various digital techniques and specific features of the Xilinx Virtex-IIV2MB1000 FPGA to accelerate operations.

A wide range of colors can be created by the primary colors red, blue, and yellow, if working with paints. Those colors then define a color space. We can specify the amount of red color as the X

axis, the amount of blue as the Y axis, and the amount of yellow as the Z axis, giving us a three-dimensional space, wherein every possible color has a unique position.

However, this is not the only color space. For instance, when colors are displayed on a computer monitor, they are usually defined in the RGB (red, green and blue) color space. This is another way of making the same colors, and red, green and blue can be considered as the X, Y and Z axes. Another way of making the same colors is to use their hue (X axis), their saturation (Y axis) and their brightness (Z axis). This is called the HSB color space. There are many more color spaces. Many can be represented as three-dimensional (X,Y,Z) values in this way, but some have more, or fewer dimensions, and some cannot be represented in this way at all.

The RGB color model is implemented in different ways, depending on the capabilities of the system used. By far the most common general-use incarnation as of 2006 is the 24-bit implementation, with 8 bits, or 256 discrete levels of color per channel. Any color space based on such a 24-bit RGB model is thus limited to a gamut of 256×256×256 ≈ 16.7 million colors. Some implementations use 16 bits per component, resulting in the same range with a greater density of distinct colors.

This is especially important when working with wide-gamut color spaces (where most of the more common colors are located relatively close together), or when a large number of digital filtering algorithms are used consecutively. The same principle applies for any color spaces based on the same color model, but implemented in different bit depths.

The RGB representation uses additive color mixing, Fig.1, because it describes what kind of light needs to be emitted to produce a given color. Light is added together to create form out of the darkness. RGB stores individual values for red, green and blue:

$$\mathbf{C} = r\mathbf{R} + g\mathbf{G} + b\mathbf{B}$$

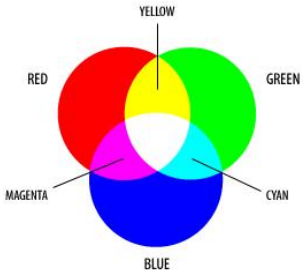


Fig.1: RGB Color Mixing

2 HSL (hue, saturation, lightness/ luminance)

Also known as HSI (hue, saturation, intensity) is close to the human perception of colors. The main advantage of this representation is its capacity to ensure a decorrelation of color information. For example, the colors “red” and “pink” are distinguished only based on the saturation component, whereas in the RGB space, there is a high correlation between the three components [1][2]. The HSL color space has three coordinates: hue, saturation, and lightness (sometimes luminance) respectively. The hue is an angle from 0 to 360 degrees; typically 0 is red, 60 degrees yellow, 120 degrees green, 180 degrees cyan, 240 degrees blue, and 300 degrees magenta. Saturation typically ranges from 0 to 1 (sometimes 0 to 100%) and defines how gray the color is, 0 indicates gray and 1 is the pure primary color. Lightness is intuitively what its name indicates; varying the lightness reduces the values of the primary colors while keeping them in the same ratio. If the color space is represented by disks of varying lightness then the hue and saturation are the equivalent to polar coordinates (r, theta) of any point in the plane, Fig.2.

2.1 RGB to HSL Transformation:

The HSI components are obtained from the RGB ones, Fig.3, by using an algebraic transformation [1]: let Y denote the intensity and (C1, C2) the chrominance components, so:

$$\begin{bmatrix} Y \\ C_1 \\ C_2 \end{bmatrix} = \begin{bmatrix} 1/3 & 1/3 & 1/3 \\ 1 & -1/2 & -1/2 \\ 0 & \sqrt{3}/2 & -\sqrt{3}/2 \end{bmatrix} \begin{bmatrix} R \\ G \\ B \end{bmatrix}$$

The Hue, Saturation and Intensity components are then given by the following equations:

$$I = Y = (R+G+B)/3, \quad S = \sqrt{C1^2 + C2^2}$$

$$C3 = atan2(C1,C2)$$

$$If\ (C3>\pi/2)$$

$$Hue=((\pi/2-C3)+2*\pi)$$

$$else$$

$$Hue=(\pi/2-C3);$$

Because Hue component has a cyclic value and so it can be represented by the following Fig.2:

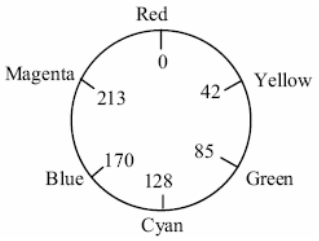


Fig.2: The circular representation of the Hue component

Saturation is a percentage value and describes the amount of grey which contributes to a color. Thus a value of 0% results in a plain grey whereas a value of 100% describes a pure color.

Lightness is also a percentage value. A lightness of 0% results in Black and a lightness of 100% results in White, both regardless of hue or saturation.

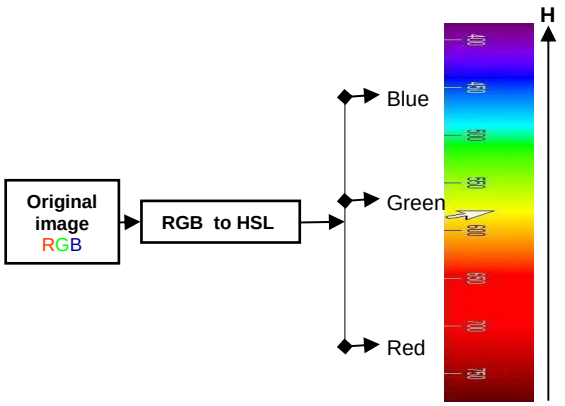


Fig.3: Color transformation from RGB to HSL

Color detection in color images is highly useful in many applications including image robotics, target recognition, image content-based retrieval….etc. Our flexible implementation utilizes various digital techniques and specific features of the Xilinx Virtex-IIV2MB1000 FPGA to accelerate operations.

Our approach in this application is choosing one color to be detected and the rest of colors are maximized to the white color (255). The selected color represents the region of interest (RI).

2.2 The Algorithm

1) Start.
2) Image conversion from RGB space to HSL space: each color is characterized by H-component within an interval.
3) In order to isolate colors, we apply hysteresis thresholding using two limit values for each color (for example: the blue color interval is [-π/2,+ π/2]).
4) If the pixel color belongs to the thresholding interval, we affect it a defined color.
5) If not, we affect to this pixel a black or white color.
6) End.

3 Concept of reconfigurable computing:

The advance in technology has made possible to produce better programmable circuits with FPGAs and with increased performance and higher integration densities. FPGAs offer the highest flexibility and cost effectiveness because hardware implementation can be done several times on the same chip at no additional cost and because mass productions of FPGA chips have lead to very competitive prices.

Oriented Application Specific Integrated Circuits (ASICs), can perform a restricted set of operations/tasks, but configurable computing hardware, such as Field Programmable Gate Arrays (FPGAs), allow modifications at any stage of the design.

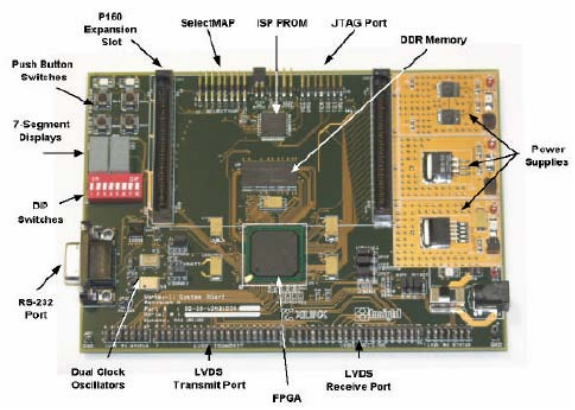


Fig.4: Virtex-IIV2MB1000 System Board

3.1 EDK: Embedded Development Kit

The EDK is a development environment that provides application designers with the tools necessary to build embedded soft cores processor systems. The steps within the EDK that are necessary to build the embedded processor system include: hardware platform creation, software platform creation, and software application creation. The hardware platform is defined by the Microprocessor Hardware Specification (MHS) file which defines our system architecture, memory modules, embedded processors and the systems connectivity as well as the configurable options and the address map for each memory module in our system.

The Xilinx Platform Studio (XPS) which is an integrated development environment gives a wide selection of settings to modify the kernel as needed. It allows the user to, for example, initiate interrupt handlers, add thread support, and specify clock resolution. All modifiable settings can be seen in Xilinx OS and libraries documents.

3.2 The MicroBlaze Processor:

The Virtex-II FPGA in addition to containing the traditional elements that are characteristics of previous platform FPGA generations contains the MicroBlaze Soft Core Processor Block [9].

The Microblaze (MB) is an embedded soft processor system designed by Xilinx as an intellectual property (IP) core that is implemented using the logic primitives of the FPGA. This processor provides application designers with the tools needed to build embedded soft cores processor systems. The architecture is shown below in Fig 5.

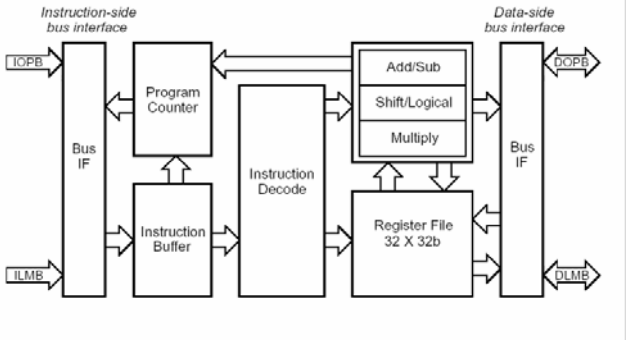


Fig.5: Microblaze Architecture

It has the following features:

- RISC Processor.
- Thirty-two 32-bit general purpose registers.
- 32-bit instruction word with three operands and two addressing modes.
- Separate 32-bit instruction and data buses OPB (**O**n-chip **P**eripheral **B**us).
- Separate 32-bit instruction and data buses LMB (**L**ocal **M**emory **B**us).
- Hardware multiplier (in Virtex-II and subsequent devices).

Embedded systems design is a hot application field which merges logic design and processor-based hardware development in a single or few chips solution. In recent years, embedded applications have emerged at a fast rate and used in every field. Various technologies have been used in the development of embedded systems; microcontroller, DSP processor, ASIC, and now FPGA.

4 FPGA implementation of the Algorithm

The original source code was written in standard C language [6] and was not optimized for any specific processor target. The algorithm was designed to detect one of the three basic colors: Red, Blue and Green in color images of any size MxN.

The change made to the C code in support of hardware compilation is how to load the input image data to the memory and how to read the processed output image by the host computer. The

algorithm assumes the data in a global matrix (image).

4.1 Platform creation

After simulating its functionality using standard desktop tools, we are ready to implement the application on a mixed FPGA/processor target. We choose a Xilinx MicroBlaze based FPGA target, selecting the Virtex-II MicroBlaze Development Kit (EDK) as our reference system. This kit includes a hardware reference board populated with a VirtexII FPGA and various peripheral interfaces, as well as all development tools needed to compile and synthesize hardware and software applications (consisting of HDL source files for hardware and C source files for software) to the FPGA target. When combined with the C compiler, this kit provided us with everything needed to compile and execute our application from our C language source files.

Microblaze processor provides four bus connections, namely the local memory bus (LMB), the on-chip peripheral bus (OPB), the fast simplex link (FSL) and the xilinx cachlink (XCL). The LMB is a dedicated bus for MicroBlaze and on-chip block RAM connection. The OPB is a CoreConnect IBM standard bus and gives the capability to connect variety of available IP blocks and peripherals, Fig.4. The FSL has a FIFO-based interface and it provides a connection between a custom hardware to assist particular application. Lastly the XCL interface provides a link between MicroBlaze processor and data and instruction caches [3].

The following steps were required to create our reconfigurable platform hardware:

1- Platform creation : Microblaze + UART+ DDRAM + GPIO + MDM.

2- Write a small program to be executed by the microblaze on the BRAM : the microblaze boots from the BRAM. The created compiled program is : executable.elf.

3- Write the C program for segmentation (add C source), compiled under the name: executable1.elf.

4- Execution :

4 -1- Load the executable main program using the EDK procedures and configure FPGA.

4 - 2- Run the XMD window, and go to the directory : mblaze and the subdirectory : code.

4 - 3- Connect XMD to the microblaze using : connect mb mdm (mdm: microprocessor debug module).

4 – 4 - Load the original image (data) using : xdownload 0 –data image.* 0x8d000000.

4- 5- Load the encoding program using : Xdownload 0 executable1.elf.

4 - 6- Run the hyper terminal.

4 -7- Run the command : text capture, in the hyper terminal and wait.

4- 8- Run the compiled C-programe using the command: xcontinue 0 0x8c000000.

4- 9- Output data is transmitted through the serial port to the hyperterminal.

4-10- When completed, stop text capture, data is written in file by the host PC.

Through this work, we have created a mixed hardware/software application, Fig.5. The software portion communicates directly with the hardware process using streams-based communications. We have shown the steps needed to generate the FPGA hardware [4][5][9]. Combine this with a soft processor and download the combined application to an actual FPGA prototyping board.

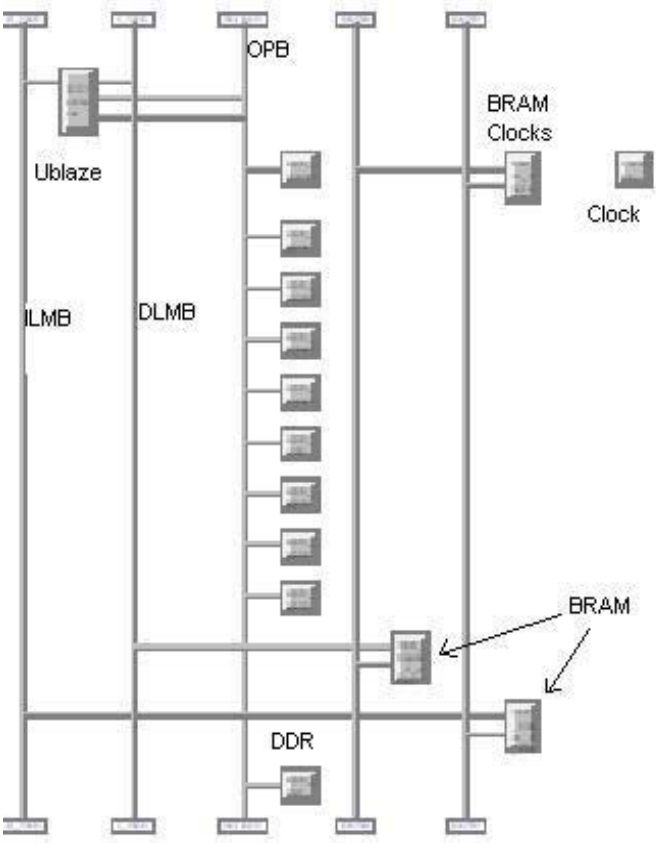


Fig.6: The created hardware plate form

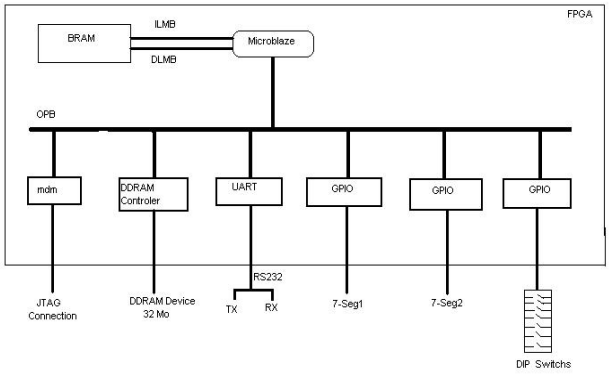


Fig.7: Design block diagram

4.2 Compiling to Hardware

The created platform, with MicroBlaze and peripherals, lets us validate the algorithm quickly and apply a wide variety of input programs simply by changing and recompiling the software application [7], which runs on the FPGA's embedded processor.

Armed with this software/hardware implementation, we can now proceed to optimize this application for performance, secure in the knowledge that the results will be fully testable at the process level, in actual hardware.

5 Test results

Different types of images are used to test the implementation, ranging from simple texts, Fig.8,9,10. human faces and natural scenes, medical images, robotics neutron images, Fig.11,12,13,14,15,20,21. And Simple object location to high quality images, Fig.16, 17,18,19.

Acquired original images are converted to memory bit files and vice versa, initial data to be processed is written inside the memory and the result (one color image) is read back from the UART (serial port):

1- Load image file in the memory and read its pixels.

2- Execute the chosen color code from its memory location.

3- Write the one color image: transmit resulted characters towards the UART, these characters are captured by the hyper terminal and written in a file.

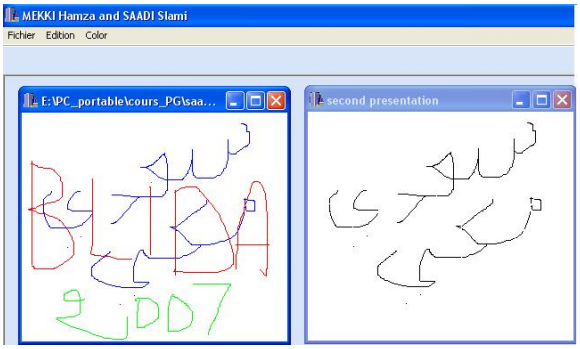


Fig.8: Extracting only blue text

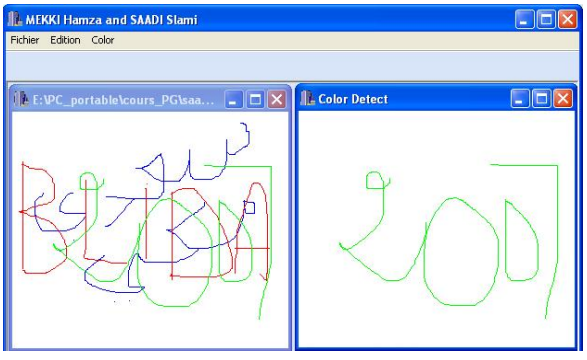


Fig.9: Extracting only green text

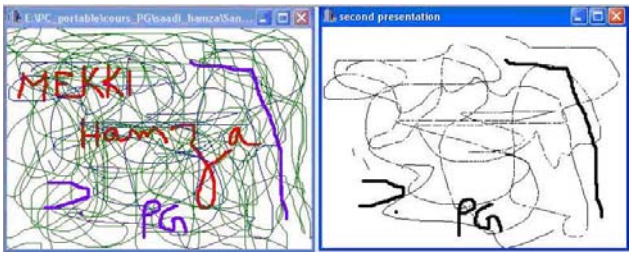


Fig.10: Extracting only blue color

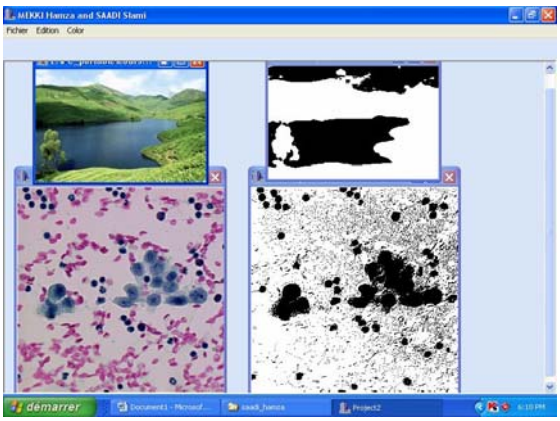


Fig.11: Original and segmented natural and cells images

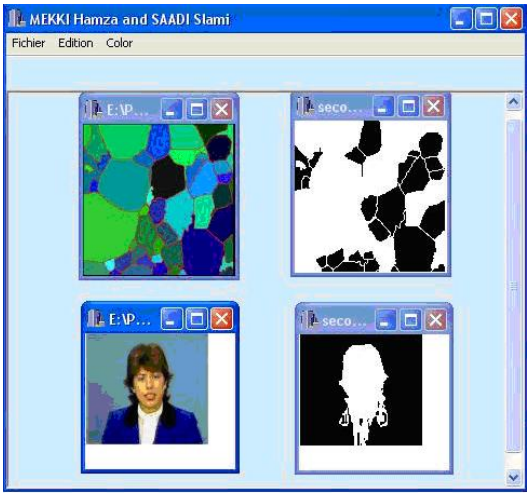


Fig.12: Original and segmented color and face images

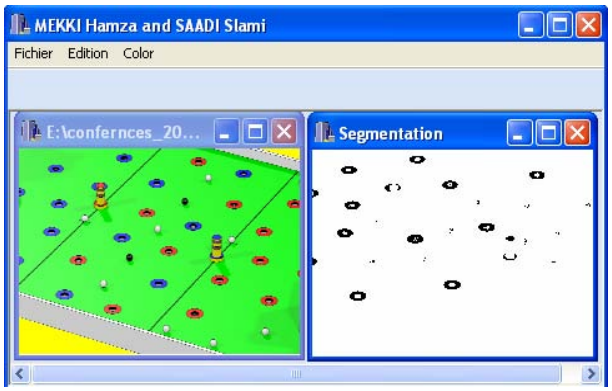


Fig.13: Original and segmented image for robotics

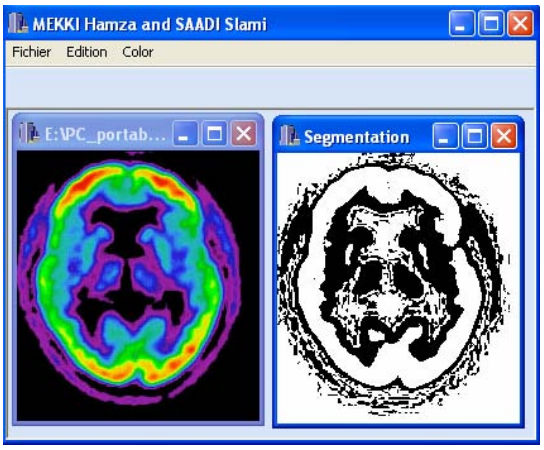


Fig.14: Original and segmented MRI image

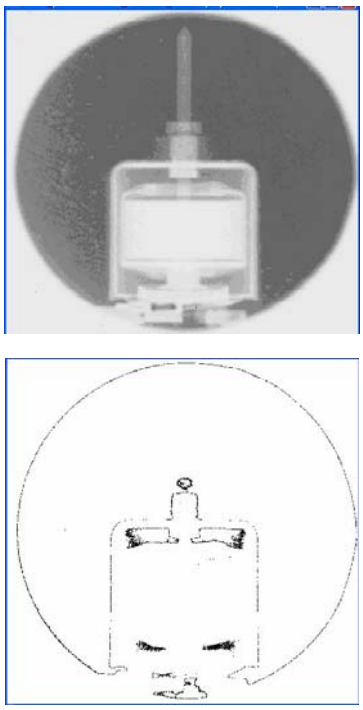


Fig.15: Original and segmented Neutron image

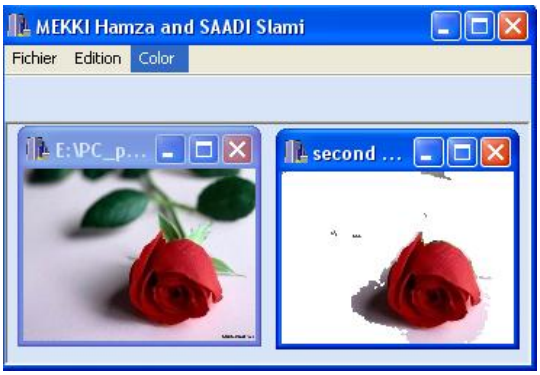


Fig.16: Extracting red flower

Fig.17: Detecting merely green color

Fig.18: Original and detected green elements

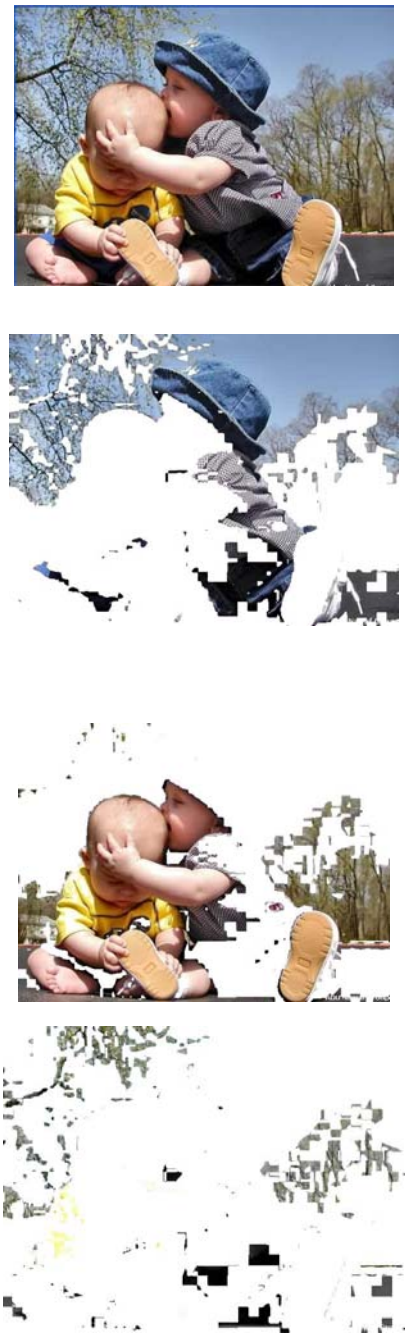
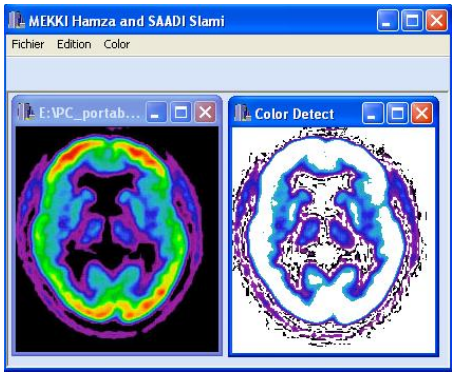


Fig.19: Original image and the three detected levels of colors



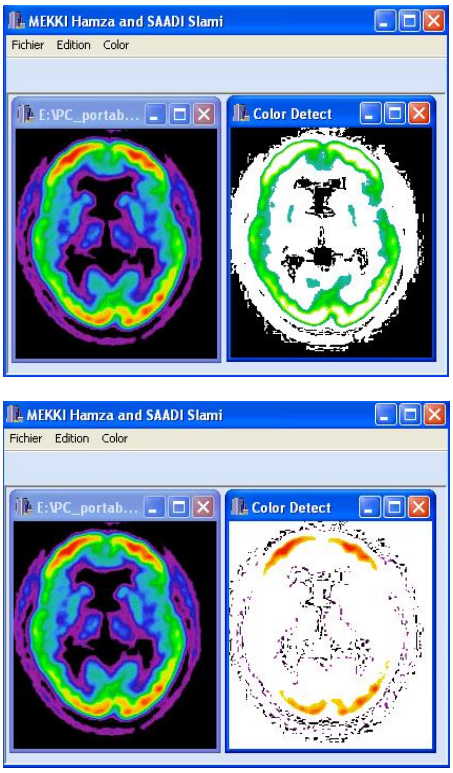


Fig.20: Three basic colors detected for a PET image

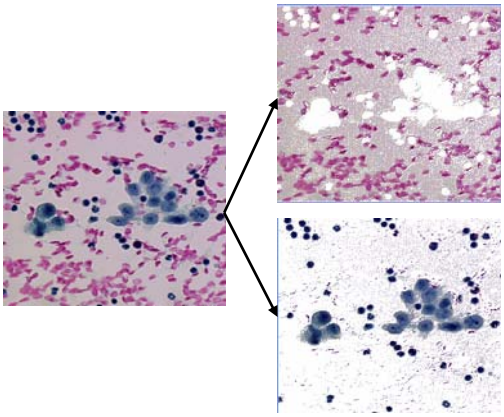


Fig.21: Extracting Cancer Cells (Detecting Blue Color)

Timing is an important metric when comparing the hardware and software implementation. The timing results of our algorithm implemented on Xilinx Vertex-II FPGA (frequency:100MHz) and on Pentium IV (frequency:3300MHz); indicate that for different size type images, execution time is more better on the MicroBlaze processor in addition to the embedding and low cost characteristics of the FPGA which make it the better choice for our application.

Finding a compromise between execution time and device usage is very important in hardware implementations, thus reaching up to 10% of device usage with a very short time for execution (less than milliseconds) is a very good job, suppose that the FPGA board is totally dedicated to this application. The hardware resources are summarized in Table1, in which we can see that the hardware occupation ratio of each block is less than 5% of the device.

Table.I: Hardware resources

	MicroBlaze with Hardware
Slice Flip Flops	214 out of 10,240 (2%)
4 input LUTs	295 out of 10,240 (2.8%)
Slices	206out of 5,120 (4%)
Dual Port RAMs	4

6 Conclusion

In this work, we have implemented a fast algorithm for region color detection in color images using color images segmentation on the MicroBlaze softcore processor based re-programmable FPGA.

The flexible system has multiple configurations which support different color choices, depending on the user requests. Lessons gained from this work will help us enhance similar implementations in the future for further improvements using the same reconfigurable board, especially when the whole imaging system will be achieved.

References

[1] Souhila Guerfi, Jean-Pierre Gambotto, Sylvie Lelandais, Implementation of the Watershed Method in the HIS Color Space for the Face Extraction,*0-7803-9385-6/05/IEEE*, Laboratoire Systèmes Complexes. CNRS FRE 2494. 40, Rue du Pelvoux 91020 Evry Cedex France, 2005.

[2] Marc Liévin and Franck Luthon, Nonlinear Color Space and Spatiotemporal MRF for Hierarchical Segmentation of Face Features in Video, *IEEE Transactions on Image Processing*, Vol. 13, No1, January 2004.

[3] Xilinx, January 2002, *Virtex II Pro Platform FPGA Handbook*, Ver1.0 (available at www.xilinx.com).

[4]www.xilinx.com/ise/embedded/edkdocs.htm.

[5] J.Viejo, M.J. Bellido, Efficient Design and Implementation on FPGA of a MicroBlaze Peripheral for Processing Direct Electrical Networks Measurements,1- 4244–0777-X/06.IEEE, 2006.

[6] David Pellerin, Scott Thibault, Pub Date: April 22,2005, *Practical FPGA Programming in C*, Prentice Hall PTR, ISBN: 0-13- 154318-0.

[7] S.Saadi, M.Touiza and A.Guessoum, Embedded System for Image Encoder/Decoder Based on Fast Wavelet MALLAT's Algorithm: Application to Nuclear Imaging, *IEEE, 4th International Conference on Computer Integrated Manufacturing CIP'2007*, 3-4 November 2007, Setif, Algeria.

[8] A.O. Boudraa, Habib Zaidi, *Image Segmentation Techniques in Nuclear Medicine Imaging*, Division of Nuclear Medicine Geneva University Hospital Geneva,CH-1211 Switzerland.Springer Science Business Media, Inc, 2006.

[9] Hyunjin Lim, Kisun You, Wonyong Sung, 2006, *Design and Implementation of Speech Recognition on a Softcore Based FPGA*, 1-4244-0469-X/06/IEEE, School of Electrical Engineering, Seoul National University, Korea.