ijecs open access www.ijecs.in

International Journal Of Engineering And Computer Science ISSN:2319-7242
Volume 3 Issue 6 June, 2014 Page No. 6544-6547

Development of Image Processing Algorithm for Array processor

Sangeeta C. Vetal [1], Prof. A.S. Shirsat [2.]

[1.] Student, ME E and TC (SP), Smt. Kashibai Navale College of Engg. Vadgaon(Bk). Pune.Maharashtra India. sangeetvetal@gmail.com

[2.] Proffessor, E and TC Department, Smt. Kashibai Navale College of Engg. Vadgaon(Bk). Pune.Maharashtra India. asshirsat@gmail.com

Abstract —***Mathematical morphology is a well known image and signal processing technique. However, most morphological tools such as Matlab are not suited for strong real-time constraints. This paper address this problem through hardware implementation on Array processor.This paper gives the algorithm of morphological image processing on FPGA/ Array Processor.***

Keywords: Image processing, Matlab, Array Processor

I. INTRODUCTION

Digital image processing is an ever expanding and dynamic area with applications reaching out into our everyday life such as medicine, space exploration, surveillance, authentication, automated industry inspection and many more areas. Applications such as these involve different processes like image enhancement and object detection. Implementing such applications on a general purpose computer can be easier, but not very time efficient due to additional constraints on memory and other peripheral devices.

Application specific hardware implementation offers much greater speed than a software implementation. With advances in the VLSI technology hardware implementation has become an attractive alternative. Implementing complex computation tasks on hardware and by exploiting parallelism and pipelining in algorithms yield significant reduction in execution times.

There are two types of technologies available for hardware design. Full custom hardware design also called as Application Specific Integrated Circuits (ASIC) and semi custom hardware device, which are programmable devices like Digital signal processors (DSPs) and Field Programmable Gate Arrays (FPGA's).Full custom ASIC design offers highest performance, but the complexity and the cost associated with the design is very high. The ASIC design cannot be changed and the design time is also very high. ASIC designs are used in high volume commercial applications. In addition, during design fabrication the presence of a single error renders the chip useless. DSPs are a class of hardware devices that fall somewhere between an ASIC and a PC in terms of the performance and the design complexity.

II DESIGN AND IMPLEMENTATION ALGORITHM

A.Software Algorithm

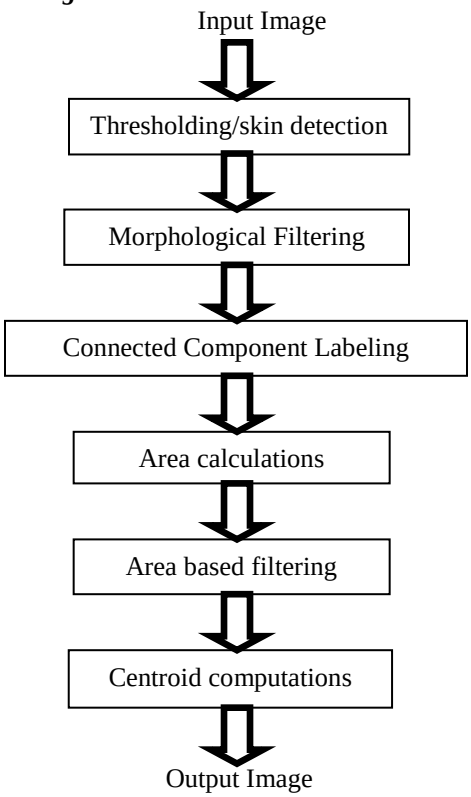

Fig.1 - Software Algorithm

B. Modified YUV color space

Converting the skin pixel information to the modified YUV color space would be more advantageous since human skin tones tend to fall within a certain range of chrominance values (i.e. U-V component), regardless of the skin type. The conversion equations are shown as follows [1].

$$Y = \frac{R+2G+B}{4}, \quad U = R - G, \quad V = B - G$$

These equations allowed thresholding to work independently of skin color intensity.

45 34 30	#2D221E
60 46 40	#3C2E28
75 57 50	#4B3932
90 69 60	#5A453C
105 80 70	#695046
120 92 80	#785C50
135 103 90	#87675A
150 114 100	#967264
165 126 110	#A57E6E
180 138 120	#B48A78
195 149 130	#C39582
210 161 140	#D2A18C
225 172 150	#E1AC96
240 184 160	#F0B8A0
255 195 170	#FFC3AA
255 206 180	#FFCEB4
255 218 190	#FFDABE
255 229 200	#FFE5C8

Fig.2 - Different skin tone samples [1]

C. Thresholding/Skin Detection

After skin pixels were converted to the modified YUV space, the skin pixels can be segmented based on the following experimented threshold.

$$10 < U < 74, \; -40 < V < 11$$

Applying the suggested threshold for the U component would produce a binary image with raw segmentation result, as depicted in Fig.3.

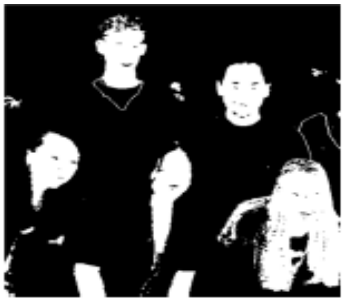

Fig.3 - Result after thresholding

D. Morphological Filtering

Realistically, there are so many other objects that have color similar to the skin color. As seen in Fig. 3, there are lots of false positives present in the raw segmentation result. Applying morphological filtering including erosion and hole filling would, firstly, reduce the background noise and, secondly, fill in missing pixels of the detected face regions, as illustrated in Fig.4.

Fig.4 - Result after morphological filtering

E. Area-Based Filtering

Human faces are of similar size and have largest area compared to other skin regions, especially the hands. Therefore, to be considered a face region, a connected group of skin pixels need to have an area of at least 26% of the largest area. Therefore, many regions of false positives could be removed in this stage, as depicted in Fig.5.

Fig.5 - Result after area-based filtering

***F*. Centroid Computation**

The final stage was to determine face location. The centroid of each connected labeled face region can be calculated by averaging the sum of X coordinates and Y coordinates separately. The centroid of each face region in Fig.6 is denoted by the blue asterisk. Here the centroid of each connected region was extracted using region props.

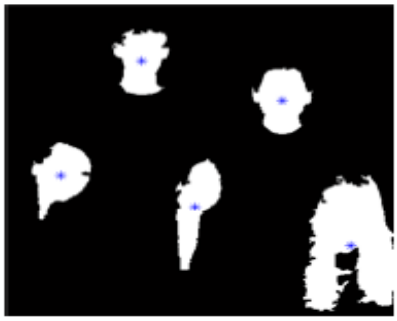

Fig. 6 - Result after calculating centroid

III. HARDWARE IMPLEMENTATION

Video Frame

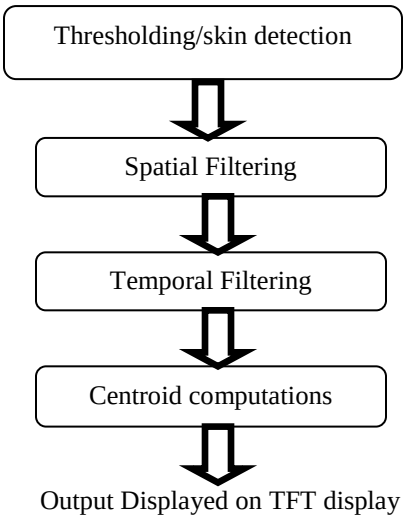

Fig. 7 - Hardware Algorithm

A. Thresholding

In this step, each input video frame was converted to a "binary image" showing the segmented raw result.

B. Spatial Filtering

This step similar to the erosion operation used in the software algorithm. However, the structuring element used here did not have any particular shape. Instead, for every pixel p,its neighboring pixels in a 9x9 neighborhood were checked. If more than 75% of its neighbors were skin pixels, p was also a skin pixel. Otherwise p is a non-skin pixel. This allowed most background noise to be removed because usually noise scattered randomly through space. In Fig. 8, because p only has 4 neighboring pixels categorized as skin, p is concluded to be a non-skin pixel and, thus, converted to a background pixel.

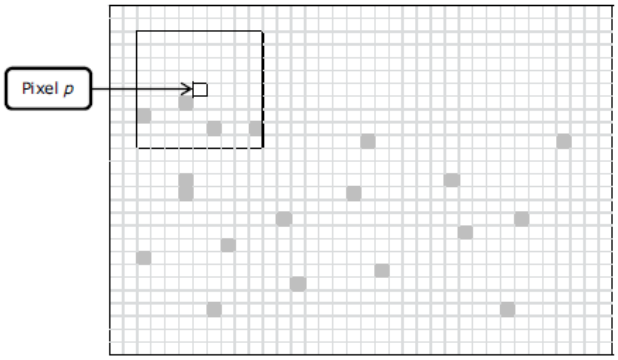

Fig. 8 – e.g of spatial filtering for a pixel p after filtering

C. Temporal Filtering

Even small changes in lighting could cause flickering and will make the result displayed on the TFT display less stable. Applying temporal filtering allowed flickering to be reduced significantly.

D.Centroid Computation

Finally, centroid is computed to locate the face region. First assume that only one face is present. Therefore, its centroid would just be the centroid of all detected pixels, as shown in Fig.9.

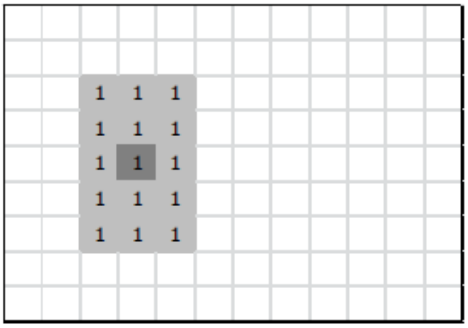

Fig.9 - Centroid of all detected pixels

IV. RESULT

Before the algorithm is developed in Quatras, Altera cyclone III FPGA , it was implemented and tested on still images in MATLAB to verify its functionality, as illustrated from Fig.3 to Fig.6. The Image Processing Toolbox provided in MATLAB allowed the process of developing and testing the algorithm to be more efficient.

V. CONCLUSION

In this paper, the goal of implementing a system to detect and track human faces in MATLAB is achieved. Although the transition from software to hardware required some modification to the original algorithm. The face detection algorithm is derived from a skin detection method. Face tracking is achieved by computing the centroid of each detected region. The system will proved to work in real time with no lagging and under varying conditions of facial expressions, skin tones, and lighting.

VI. ACKNOWLEDGMENTS

I would like to thank Prof.A.S.Shirsat for his valuable guidance. This paper would have been impossible without his guidance

VII. REFERENCES

[1] Hideaki Ito, Masaru Shimizu and Saburou Iida, "*Parallel Algorithms of Basic Image Processing Implemented on a Linearly Connected Parallel Processor- Gray-scale and Binary Images* ",Journal of Next Generation Information Technology. Volume 2, Number 2, May 2011

[2] M. Ooi, "*Hardware Implementation for Face Detection on Xilinx Virtex-II FPGA Using the Reversible Component Transformation Color Space*," in Third IEEE International Workshop on Electronic Design, Test and Applications, Washington, DC, 2006.

[3] S. Paschalakis and M. Bober, "*A Low Cost FPGA System for High Speed Face Detection and Tracking*," in Proc. IEEE International Conference on Field-Programmable Technology,Tokyo, Japan, 2003.
[4]Yates R. B., Thacker N. A., Evans S. J., Walker S. N., Ivey P. A., "*An array processor for General Purpose Digital Image Compression*", IEEE Journal of Solid State Circuits, Vol.30, No. 3, March 1995.