Indian Journal of Science and Technology, Vol 9(34), DOI: 10.17485/ijst/2016/v9i34/95610, September 2016
ISSN (Print) : 0974-6846
ISSN (Online) : 0974-5645

FPGA Specific Real Time Hardware Architecture Implementing Bounding Box Merging Algorithm for Object Detection

Bhavya Alankar[1]* and Binod Kumar Kanaujia[2]

[1]Department of Computer Science and Engineering, Hamdard University, New Delhi, India and Research Scholar, Uttrakhand Technical University (UTU), Dehradun, India; bhavya.alankar@gmail.com
[2]School of Computational and Integrative Sciences, Jawaharlal Nehru University, New Delhi –110067, Delhi, India; bkkanaujia@yahoo.co.in

Abstract

Objectives: This paper focuses on the real time hardware implementation of improved version of Viola Jones algorithm for automatic object detection. **Methods/Statistical Analysis**: In this regard, A greedy NMS(Non Maximum Suppression) approach has been adopted to develop the bounding box merging algorithm which could suppress the confusing dense grid of overlapping bounding boxes against a threshold value and gives an accurate result and further this algorithm has been transformed in to a real time hardware architecture. **Findings**: The hardware architecture designed is fully Field Programmable Gate Array (FPGA) based which will take the input coordinates of the image and will process it accordingly by reducing the number of overlapped bounding boxes in real time. **Applications/Improvements:** Any researcher who is working on automatic object detection can use this hardware architecture as it is in his design and further this architecture itself can be used to create Application Specific IC (ASIC).

Keywords: Bounding Box, Field Programmable Gate Array (FPGA), Non Maximum Suppression (NMS), Non-Maximum Suppression Finite State Machine (NMS-FSM), Real Time, Video Graphics Array (VGA)

1. Introduction

We are all aware that object tracking is a very challenging task in the presence of variability illumination condition, background motion, complex object shape, partial and full object occlusions. In spite of these difficulties, however, a lot of research has been done and progress has been made on the problem of detecting objects or pedestrians within an image in a wide field of applications such as surveillance systems, traffic assistance systems, autonomous robot navigation, video stabilization, automated vehicle parking systems, cell counting in bio-imaging etc., all of which solely depend upon the optimization of object detecting algorithms[1-3]. While many approaches have been adopted for object detection, one of the most acceptable approaches is to run a "scanning window" detector across the image at a dense grid of locations and scales, as in the Viola-Jones face detector and the pedestrian detector algorithms[1,2] These algorithms have many advantages such as a very fast and efficient feature selection, scale and location invariant detector, i.e., a generic detection scheme that can be trained for detection of other types of objects (e.g., cars, hands etc.). However, the major disadvantage of these algorithms is that such methods typically produce a number of positive responses that are close to the correct detection and this dense output leads to a confusing interpretation as far as the processing and understanding of the image is considered. This, in turn, leads to the need to have a further Non Maximum Suppression (NMS) stage to thin out and suppress the multiple spurious responses[4,5]. The most suitable approach is the greedy NMS approach. The procedure starts by selecting the best scoring bounding box and assuming that it indeed covers an object. Then, the

*Author for correspondence

bounding boxes that are too close to the selected boxes are suppressed. Out of the remaining boxes, the next top-scoring one is selected, and the procedure is repeated till a limited number of bounding boxes remain as desired by the user or the required number of boxes needed to process the image[6,7]. This procedure involves defining a measure of similarity between bounding boxes and setting a threshold for suppression.

Although a lot of research has already been done on this[8-11], we would also like to take a small step and contribute towards it further. The main aim of our work is to design an efficient real time hardware architecture of bounding box merging algorithm based on the greedy NMS approach that will take the input image that has a dense grid of overlapping bounding boxes and give the output of a minimum number of bounding boxes as desired by the user, as shown in Figure 1(a) and 1(b). The hardware architecture designed by us is based on an Field Programmable Gate Array (FPGA) based architecture and incorporates the algorithm along with the display unit that is needed for the display of the output result[3,12]. In the next section, we will describe a modified version of the bounding box merging algorithm in detail and then the FPGA based real time hardware architecture needed to implement the algorithm. Complete FPGA implementation along with simulation diagrams and resource utilization of the hardware architecture will be dealt in our upcoming work.

2. Bounding Box Merging Algorithm

This algorithm basically works on the principle of the greedy NMS approach by suppressing the bounding boxes against a threshold value as desired by the user[7,8]. The algorithm takes the coordinates of bounding boxes as input in the form of (x,y,h,w) where (x,y) are the coordinates and (h,w) are the height and width respectively. These coordinates are stored in a matrix called Temp. In addition to this, we have another matrix called Result, which stores the coordinates of the final bounding boxes left behind after the application of the algorithm. Apart from this, a few variables are declared and initialized for the smooth functioning of the algorithm. The declarations assumed and the algorithmic procedure has been explained in the subsections.

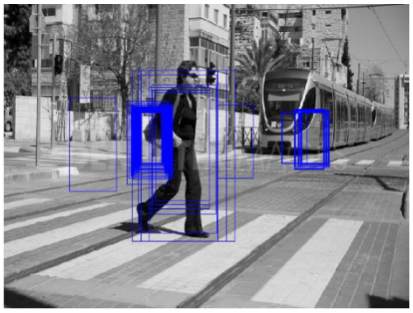

Figure 1. (a) Before applying bounding box merging algorithm

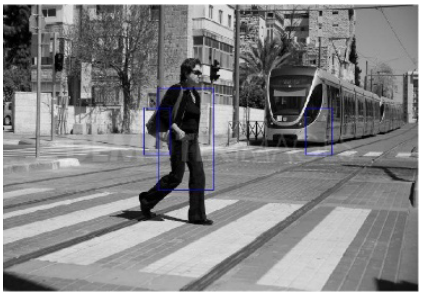

Figure 1. (b) After applying bounding box merging algorithm

Declarations:

Temp Matrix: // Coordinates of all the input bounding boxes in the form of (x,y,h,w)

N_result: // No. of coordinates in Temp matrices

Result: // Coordinates of the resulting bounding boxes obtained after running the algorithm

N_ped: // variable stores the number of bounding boxes in matrix N_result

I: // variable points to the current bounding box in Temp Matrix, where initially i=2 and N_ped =1 // and one box is inserted in N_result matrix

j: // variable points to the current bounding box in the N_result matrix. Initially j=1 as N_ped=1 //. Therefore, there is one bounding box in the N_result matrix

Flag_new: // variable is initialized to '0' if the bounding box is suppressed, otherwise it is initialized to 1.

Initially if i<N_result, then Flag_new and j are initialized to 1, which means that there is the input number of overlapping bounding boxes in the Temp matrix, otherwise the algorithm would be suspended. If this is not the case, then j is compared to N_ped. If j<= N_ped, then the distance is calculated against a minimum threshold. Again, if the distance is found to be less than the threshold, then the bounding box is suppressed and the Flag_new is initialized to 0. The value of j is then incremented by 1, and once again this is compared to N_ped

to see whether there is another bounding box in the N_ result matrix. Thus, the whole procedure may be repeated otherwise the Flag_new will be initialized to 1 and the bounding box will be added to N_result matrix i.e., (N_ped) = N_ped+1 and the result (N_ped) = Temp (i)) and I will be incremented to 1 so that it can take another bounding box from the Temp matrix for processing. This algorithm is repeated until all the bounding boxes have been processed and a limited number of non overlapping bounding boxes remain as decided by the threshold value. The whole algorithmic procedure is also depicted in the form of a flow chart in Figure 2

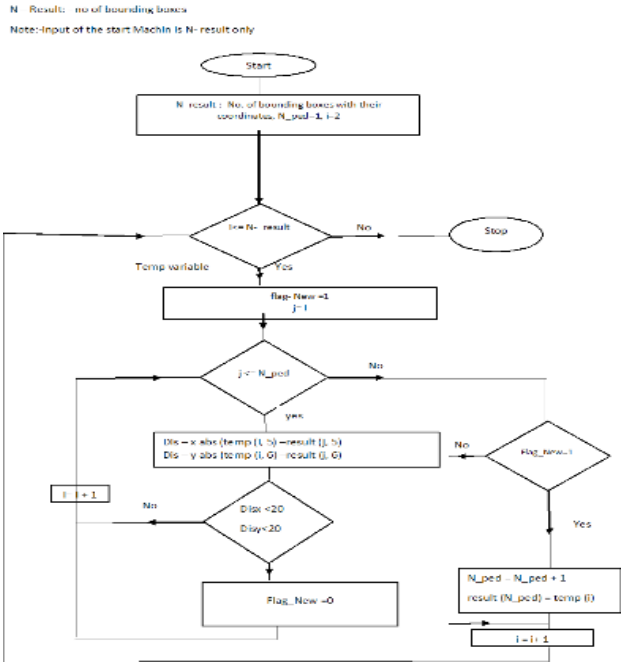

Figure 2. Flow chart representation of bounding box merging algorithm

3. Proposed FPGA Based Real Time Hardware Architecture

Before moving to the hardware architecture, let us discuss the system as a whole. Our system basically consists of three main elements, out of which one is the input coordinate of the image another one is the FPGA based real time hardware architecture while the final one is the display screen as shown in Figure 3. Now let us focus on the main element, i.e., the real time hardware architecture, which is divided into four major modules as depicted in Figure 4. The main module of this architecture is the NMS memory interaction block, which mainly consists of an FSM in which the greedy NMS approach has been implemented, as well as the memory block which consists of eight dual port memory blocks of 256 MB containing the coordinates of the bounding boxes. A detailed description of this block along with the FSM is given in the following subsection. The other modules consist of the initial display module, which is used to display all the bounding boxes in the image and the resulting display module, which is used to display the minimum resulting number of bounding boxes obtained after the application of the bounding box merging algorithm. Both these modules use the Video Graphics Array (VGA) module to display the image of the object on the screen.

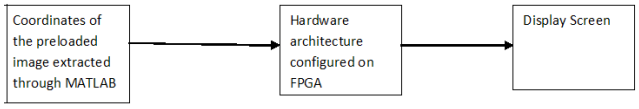

Figure 3. System level representation

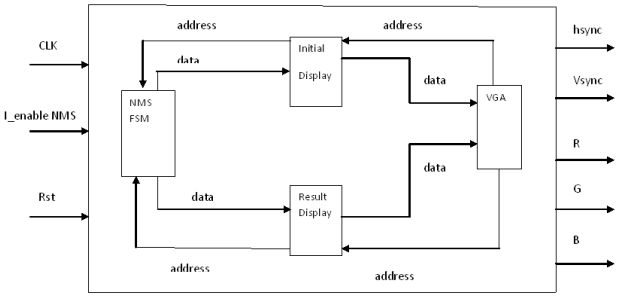

Figure 4. Hardware architecture of bounding box merging algorithm

3.1 FSM Memory Interaction Block

As discussed above, this is the most critical module of the whole hardware architecture, in which the bounding box merging algorithm is implemented as shown in Figure 5. The designing of this module involves the Non-Maximum Suppression Finite State Machine (NMS FSM), which may also be called the main controller of the hardware architecture. Interaction between the memory block and the NMS FSM block is the main driving force behind implementation of the bounding box merging algorithm. Before proceeding further, we need to briefly understand these two blocks in the following subsections.

3.1.1. Memory Block

The memory block is constituted of eight dual port memory blocks of 256 MB which are equally divided into two parts: Temp and Result. The Temp portion consists of

four memory blocks (x,y,h,w) which have coordinates of the unsuppressed and overlapping bounding boxes. These coordinates are the outcome of the Viola-Jones algorithm executed on a preloaded image in the MATLAB simulation environment as shown in Figure 3. Besides this, the Result portion also consists of four dual port memory blocks (x,y,h,w), which have coordinates of the remaining bounding boxes after the application of the bounding box merging algorithm. Inputs to the result portion are the output of the NMS FSM block.

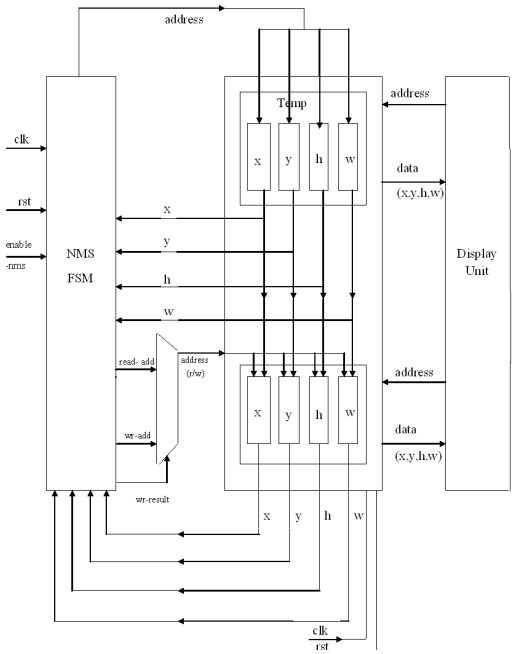

Figure 5. FSM memory interaction block

3.1.2 NMS FSM Block

As discussed earlier, this block is the most dominating block of the proposed hardware architecture in which the bounding box merging algorithm is implemented. This block uses eight bit addressing lines to fetch ten bits of data (coordinates) from the Temp portion of the memory block. The fetched data is used by the eleven states FSM as shown in figure 6 to suppress the overlapping bounding box against a threshold value as decided by the user. The whole process may be understood in the following manner.

The process begins with the idle state of the FSM, which checks whether there is any bounding box. If there isn't, then the process is suspended, otherwise the process goes to the next state after performing all the variable initializations

In the next state, the distance between the coordinates is calculated and compared. If the distance is found to be less than the threshold value (which has been decided by the user), then that bounding box is suppressed and the address is generated by the FSM to read the next coordinate from the Temp portion. Otherwise, the write address is generated and the bounding box is inserted into the Result portion of the memory as depicted in Figure 5. The wr_result acts as a select line to select the address depending upon the choice made by the FSM. This is repeated until the entire set of bounding boxes has been scanned from the Temp portion of the memory block. This whole process and the functionalities of all the eleven states are depicted in tabular form in Table 1.

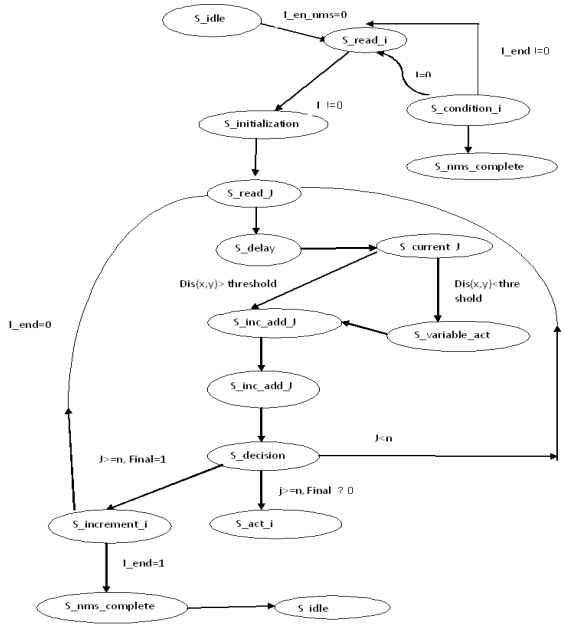

Figure 6. Finite state machine

Table 1. Finite State machine tabular representation

S_idle	According to the status of i_end,either it will goto the next state S_read_i or S_stop and will initialize the following variables as i=0,n=0,j=0 and Final =0
S_read	According to the status of i,either it will goto the next state S_condition_i or S_initialization
S_condition_i	According to the status of i_end,either it will goto the next state S_nms_complete or S_read_i and will increment i=i+1, n=n+1 and Wr_result=1
S_initialization	It will initialize j=0
S_read_j	Delay of two cycles would be given and the next state would be S_current_j_act

S_current_j_act	Distance between the coordinates would be calculated and compared and if the distance is found to be less than some threshold value then the next state would be S_variable_act otherwise the next state would be S_inc_address_j
S_variable_act	The next state would be S_inc_address_j and Final=1, O_wr_result=1
S_inc_address_j	The next state would be S_decision and J would be incremented as j=j+1

S_decision	Three conditions would be there
	If j>= n and Final=0 then
	Next state would be S_act_i
	If j>= n and Final=1 then
	Next state would be S_increment_i
	If j<n then
	Next state would be S_read_j

S_act_i	Next state would be S_increment_i n=n+1, , O_wr_result=1
S_increment_i	If i_end=1 the S_nms_Complete else S_read_i Final=0 and i=i+1

3.1.3 Display Unit

The display unit is a combination of two separate units, viz., Initial display and Result display as depicted in Figure 4. The Initial display unit is responsible for displaying the image with overlapping bounding boxes stored in the Temp portion of the memory while the Result display unit is responsible for displaying the image with the suppressed bounding box stored in the Result portion of the memory after the application of the algorithm. Both of these units interact with the VGA, which is part of the hardware architecture and implemented along with the other units[13,14]. This VGA unit generates the required signals which are needed to display the image and these signals are accordingly sent to the display device.

4. Conclusions

Several researchers have used the greedy NMS approach in application specific ways for creating different applications for object detection by implementing the algorithm on different platforms such as MATLAB in order to judge the accuracy and efficiency of the algorithm. However, thus far, no one has given an insight into the real time hardware implementation of this algorithm. We have taken the greedy NMS approach in order to create a modified version of the algorithm named the bounding box merging algorithm, specially for creating the FPGA specific real time hardware architecture that can process the image immediately as it gets the input coordinates. This architecture can be used as it is by researchers, who are working in the field of object detection and want to process the image for different applications. Moreover, this architecture can be further exploited for creating ASICs

5. References

1. Viola P, Jones M. Rapid object detection using a boosted cascade of simple features. In: IEEE Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition, CVPR'01, 2001, 1, p. 511-18.
2. Viola P, Jones MJ. Robust real-time face detection. International Journalof Computer Vision. 2012; 57(2):137-54.
3. Raveendra Reddy S, Sakthivel SM. A FPGA Implementation of Dual Images based Reversible Data Hiding Technique using LSB Matching with Pipelining, Indian Journal of science and Technology. 2015; 8(25):1-6.
4. Wang W, Yi-Qing Y. An analysis of the Viola-Jones face detection algorithm. Image Processing On Line, 2014, 4, p. 128-48.
5. Neubeck A, Van Gool L. Efficient non-maximum suppression. In: IEEE 18th International Conference on Pattern Recognition, Hong Kong, ICPR'06, 2006, 3, p. 850-55.
6. Pirsiavash H, Ramanan D, Fowlkes CC. Globally-optimal greedy algorithms for tracking a variable number of objects. In: 2011 IEEE International Conference onComputer Vision and Pattern Recognition(CVPR), RI, 2011, p. 1201-208.
7. Rothe R, Rasmus R, Guillaumin M, Gool LV. Non-maximum suppression for object detection by passing messages between windows. In: Asian Conference on Computer Vision (ACCV), Springer International Publishing: Switerzerland, 2014, 9003, p. 290-306.
8. Hefenbrock D, Oberg J, Thanh NTN, et al.,. Accelerating Viola-Jones face detection to FPGA-level using GPUs. In: 18th IEEE Annual International Symposium on Field-Programmable Custom Computing Machines, Charlotte, NC, 2010, p. 11-18.
9. Brookshire B, Jon J, Steff Jorgenensen S, Xiao J. In: FPGA-based Pedestrian Detection, Barcelona, 2010, p. 530-537.
10. Hahnle M, Saxen F, Hisung M, Brunsmann U. FPGA-based real-time pedestrian detection on high-resolution images. In: 2013 Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops, Portland, OR, CVPRW'13, 2013, p. 629-35.
11. Diederichs C, Fatikow S. FPGA-based object detection and motion tracking in micro-and nanorobotics. In: Nanotechnology: Concepts, Methodologies, Tools, and

Applications: Concepts, Methodologies, Tools, and Applications, 2014, p. 251-53.

12. Kadali KS, Rajaji L. FPGA and ASIC Implementation of Systolic Arrays for the Design of Optimized Median Filter in Digital Image Processing Applications, Indian Journal of Science and Technology. 2014 Nov; 7(S7):99-103.
13. Zhang Z, Wen-ai W, Zhang B, Cheng YQ. The implementation of VGA display controller with high resolution based on FPGA [J]. Advanced Display. 2006; 9(1):13.
14. Ya-Ping Z, Zhan-Zhuang H. Design of VGA display module based on FPGA. Computer Technology and Development, 2007; 17(6):242-45.