Multiresolution Sensors with Adaptive Structure

Pelegrín Camacho, Fabián Arrebola, Francisco Sandoval
E.T.S. Ingenieros de Telecomunicación
Universidad de Málaga - Campus de Teatinos,
29071 – Málaga, Spain
pelegrin@dte.uma.es

Abstract – **This paper describes the architecture of adaptive space-variant sensors with Cartesian topologies. Besides their multiresolution output, reconfigurable sensors can be upgraded to generate additional data to be processed at higher level modules of the vision systems, making it possible to unload the processing stages of certain tasks, without penalty in time or significant addition of hardware. A synthesizable implementation of these sensors, based on off-the-shelf FPGAs and CCD cameras is also described.**

I. INTRODUCTION

There are many active vision systems requiring wide field of view -FOV-, high resolution capability and speed of response. To match those requirements, several research groups developed spatially variant or multiresolution sensors with polar geometries, imitating the human eye retinotopology [1][2][3]. Similarly, considering a trade-off between nature emulation an easiness of implementation, Bandera and Scott [4], proposed Cartesian topologies with a central fovea surrounded by concentric rings.

The advantages of multiresolution sensing, i.e., its selective data reduction in space, resolution and time, are obtained at the expense of gaze fixation mechanisms to place the fovea onto regions of interest, considering both the position and the size of the objects. Therefore, in the development of multiresolution sensors, it must be considered the number, size and placement of the elements on the sensor, since they will determine the acuity of the resolution cells, the levels of the sensor and its data compression. In turn, the acuity profile or dimensional ratio among resolution levels, in relation to the size and singularities of the objects, could require multiple foveations for recognition, leading to complex gaze control when sensors of fixed topologies are used. To overcome these problems, expensive and precise mechanical systems are required in those application where target motion and processing speed must be considered.

Within the different architectures of vision systems [5], our approach to space-variant or foveal sensors, has been oriented to develop reconfigurable structures to minimize data volume, adapting the sensor structures to the size and position of the targets, emulating saccadic eye movements with an electronic fovea [6]. Thus, mechanical systems would only be required for coarse fixation of the field of view, without the requirement of precise pointing.

II. ADAPTIVE FOVEA SENSORS

Taking profit of the characteristics available in CCD cameras and the reconfigurability of FPGAs, to emulate human vision we define variable non-concentric structures for rectangular FOVs of HxV pixels. As shown in Fig. 1, five parameters configure them:

m : Number of rings surrounding the fovea.

$$l: \text{ Left factor } \quad l = INT\left[\frac{X_{min}}{J_0}\right]$$

$$r: \text{ Right factor } \quad r = INT\left[\frac{H - X_{max}}{J_0}\right]$$

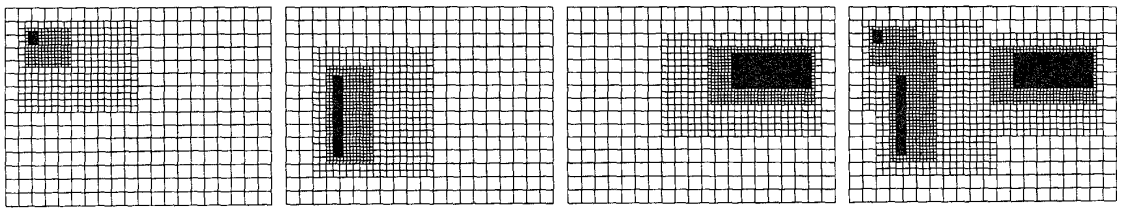
$$t: \text{ Top factor } \quad t = INT\left[\frac{Y_{min}}{J_0}\right]$$

$$b: \text{ Bottom factor } \quad b = INT\left[\frac{V - Y_{max}}{J_0}\right] \qquad (1)$$

where X_{min} , Y_{min} , X_{max} , Y_{max} refer to the coordinates of the bounding box of the target or region of interest -ROI-, and J_0 being the parameter determining the eccentricity of the foveal image mapped onto the FOV. Eccentricity is obtained after a discrete number of minimum jumps. The minimum jump, J_N , for any level N on the structure, can be defined as the minimum shift of that level across the FOV, caused by the simultaneous minimum shift of all rings on the FOV [7]. The jump, in pixels, is given by

$$J_N = 2^{N+1} + 2^{N+2} + \ldots + 2^m = \sum_{N}^{m-1} 2^{i+1} \qquad (2)$$

valid for all rings, except the peripheral one.



a) l, r, t, b = 1, 10, 1, 7 b) l, r, t, b = 2, 9, 3, 2 c) l, r, t, b = 7, 1, 2, 5 d) Merged structure of a+b+c

Fig. 1. Adaptive structures with three rings (m = 3)

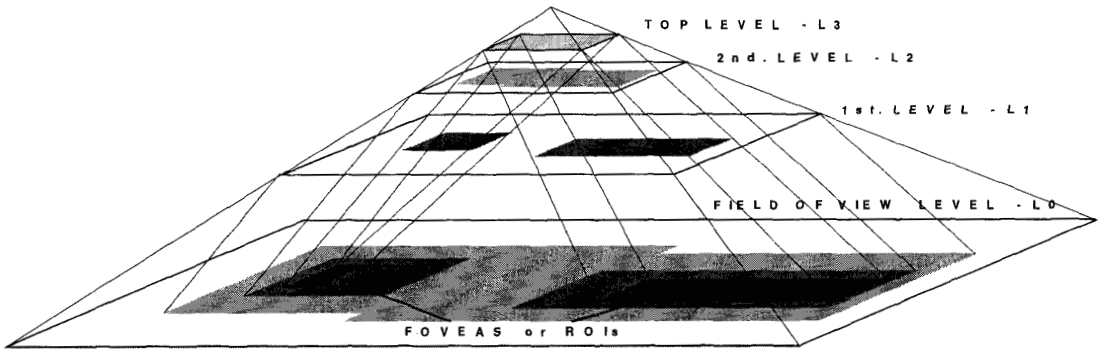


Fig. 2. Merged pyramids for an adaptive foveal structure with two foveas and three resolution levels (m= 3)

Sensors reconfiguration allows the placement of fovea and rings around areas of interest without mechanical actions. In this way, multiresolution structures are obtained with progressive eccentricity of their levels over the FOV where they are mapped.

The techniques to obtain these multiresolution structures follow the procedures used in vision pyramids, sketched in Fig. 2: departing from the uniform resolution images supplied by a CCD camera, successive 4-to-1 pixel cells averagings are performed, obtaining the resolution cells or *rexels* of the upper levels, where the resolution and data volume are progressively reduced [8][9]. To further reduce the data volume and the time required to its processing, only selected areas around the targets are considered for data processing. However, to satisfy the wide FOV requirement of active vision, regions outside these areas are considered at a lower resolution, keeping their capacity for detection, but not for recognition. After obtaining subdivision factors, all inner rings and fovea can be referred to the origin or upper left corner of the FOV. The origin, O, and the opposite corner, Z, of any level N are thus related to the jump, J_N , and their coordinates given by

$$\begin{aligned} O_{xN} &= J_N.l & O_{yN} &= J_N.t \\ Z_{xN} &= H - J_N.r & Z_{yN} &= V - J_N.b \end{aligned} \qquad (3)$$

The data structure for adaptive sensors is obtained as shown in Fig. 2. Projection of top level onto the ROIs in the FOV, performed through inner truncated pyramids, will determine the size and location of foveas in relation to the projecting angles at each side of their inner pyramids, as a result of applying the subdivision factors given in (1). Parallel projections of lower levels will determine the lateral resolution gradients and the hierarchical data structure of the sensor. This structure can be repeated for several regions of interest –ROI- originating the multifoveal structures of Fig.1d. The main advantage is the possibility of parallel processing for several ROIs. Fig. 3 shows a bifoveal image mapped into the FOV, whose resolution levels are shown in Figs. 4 and 5.

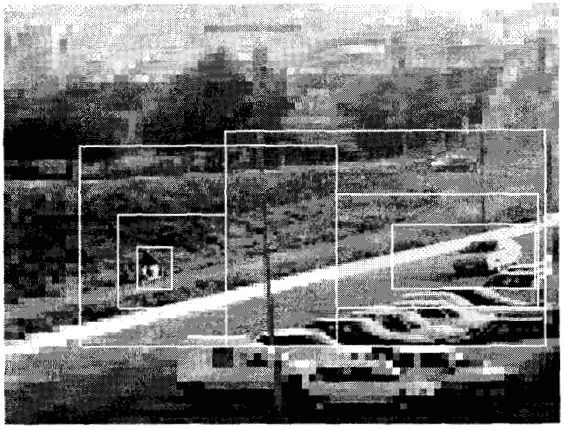
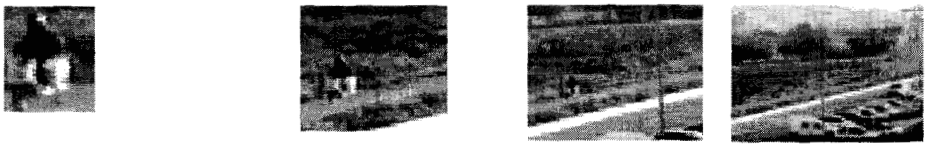


Fig. 3 Bifoveal image with three resolution rings (640 x 480 pixels)

Fovea L_0: 164 x 74 L_1 : 116 x 66 L_2 : 92 x 62 L_3 : 80 x 60

Fig. 4 Adaptive fovea levels for m = 3, obtained with l,r,t,b = 32, 2, 18, 11 (Expanded ~ 3 in relation to Fig. 3)



Fovea L_0 : 38 x 46 L_1 : 62 x 54 L_2 : 74 x 58 L_3 : 80 x 60

Fig. 5 Adaptive fovea levels for m = 3, obtained with l,r,t,b = 11, 32, 20, 11 (Expanded ~ 3 in relation to Fig. 3)

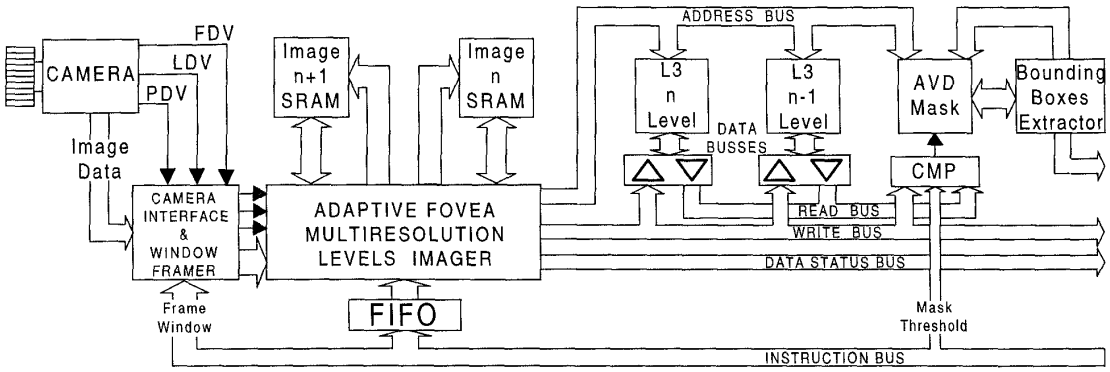


Fig. 6 Adaptive sensor subsystem incorporating modules for bounding boxes extraction

III. ADAPTIVE SENSORS IMPLEMENTATION

The first system requirement is a set of digital data inputs. However, the most common output signal of commercial CCD cameras is a composite video type, according to NTSC or CCIR standards. To reduce the hardware of the levels imager and to rely on a high quality signal source as the one supplied by digital cameras, we use a Pulnix TM9701 progressive scan CCD camera, with RS-422 outputs, giving a video rate of 30 frames/sec. and 768 x 484 pixels/frames.

The adaptive sensor architecture is shown in Fig. 6, in relation to the camera interface. Besides the byte-wide data output, the camera provides two signals for frame and line synchronism -FDV, LDV- as well as a PDV signal to validate pixels bytes. Although different types of lenses can be used to adapt the camera to the FOV requirement of potential applications by fixing the viewing angle, i.e. the optical FOV, we have considered the possibility of electronic FOV reduction to decrease the data volume by windowing on the optical image projected by the lenses onto the CCD sensor. This is done just after the RS422 level conversion performed at the camera interface. The window framer is configured as a dual programmable length counter, fixing the upper-left and lower-right window corners onto the 768 x 484 frames sent by the camera. Pixels within the window will be passed to the imager and new PDV and LDV signals are generated according to window size. Further data reduction could be fixed by controlling FDV. One out of N incoming frames can be selected with an additional modulo-N frame counter. Thus, the camera frame rate - 30 fr/sec.- can be reduced to 15, 10 or even less. This alternative can be used when the processing speed of low to medium cost vision systems is unable to process images at camera frame rate. Obviously, this could be a severe constraint when scenes contain mobiles, requiring higher processing speed.

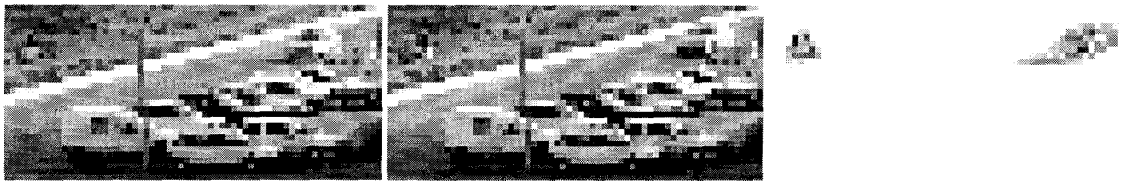
Thus, the interface can perform a selective data reduction in time and space, without affecting to the image resolution, whose control is left for the imager core. The imager subsystem, including the window frame module, is based on Altera's FLEX 81188 FPGA. For operation at full video rate and 640x480 frames, we use two 512Kx8 SRAMs in *double buffering*: while odd image *n+1* is received and stored in SRAM *n+1*, even image *n* previously stored in *n* SRAM will be processed to obtain resolution levels corresponding to the structure configured on the imager. All gaze control is left to the vision system.

Addresses of corner pixels for each level, given in (3), are processed in the host and transmitted to the imager at any time using the Instruction bus and stored in a 64x16 FIFO, from which the imager updates the corner references after processing every image. Resolution levels are obtained in a sequential process controlled with a 21-states FSM. The first level to read from image SRAMs is L_0. Selected pixels of foveas are read and sent to host through the Write Bus, while on the Data Status Bus some bits are set to identify the ROIs and levels being sent. After that, the first 4-to-1 cell averaging starts, reading 4 cells in Image *n* SRAM, adding them and shifting 2 bits to average the result. Since three averagings are done to get the successive pyramid levels (m = 3), averages are written back into the first cell whose content was used to obtain the average, implying the 4-to-1 image compression to get the next level *rexels*. After each averaging, all ROIs on the obtained level -L_1 and L_2 of Figs. 4 and 5-, are sent as the foveas were before. The procedure allows multilevel processing of several ROIs at the imager with just one set of levels averagings - the most time consuming process-, which is particularly important for scenes where several mobiles could be present.

IV. MOBILES DETECTION

The 64-to-1 compressed FOV -pyramid level 3- is obtained after the last averaging. The full L3 is sent to host and, instead of saving it back into the SRAM, it is written into *L3 n SRAM*, while the content of corresponding cells in *L3 n-1 SRAM* -L3 of previous *n-1* image- are read. Common addressing of both 8Kx8 SRAMs by *L3 Address Bus* enables cell to cell comparison of consecutive L3s, through the Write and Read Busses. Because of the frame rate -30 fr/sec.- the *brightness constancy constraint* between successive images is well met. Noise or small brightness differences are cancelled after the strong low-pass filtering due to averaging. Motion detection, or *dynamic segmentation*, is based on temporal absolute value of differences -AVD- between corresponding cells on consecutive L3s. AVDs are compared to a preset threshold of few brightness units, usually below 10, computed at the host after histogramming previous L3s and sent to the CMP module via the Instruction Bus. After comparison, a binary AVD mask is obtained containing the regions where

Fig. 7 Sequence of window images (520 x 240 pixels)



a) Third resolution level -L3- of first two images in sequence of Fig. 7 b) AVD of images in a) (Threshold = 8)

Fig.8 Top pyramid levels and AVD of two consecutive images from sequence in Fig.7 (65 x 30 pixels images , expanded 8x8)

motion was detected, covering both the past and present location of the mobiles. However, AVD masks to obtain from AVD images as in Fig. 8b, are just sets of unrelated cells to be box-bounded according to the mobiles originating them, to differentiate and identify the ROIs to examine in relation to system requirements or priorities.

V. REGIONS LABELLING ALGORITHM

Regions labelling could be a time consuming process at the system level whose reduction can be implemented at the imager level. Our aim has been to avoid software tasks, integrating the region labelling process into an additional module within the imager FPGA. In spite of availability of well known labelling algorithms [10], their application at the FPGA level were complex, consuming excessive resources. We developed a labelling algorithm based on three neighbours, suitable for synthesis at the FPGA .

At mask loading, the byte-wide content of all cells in the AVD mask memory are labelled with hex 7F in bits 1 to 7. Bit0 is the mask bit (m). Cells corresponding to ROIs have m=1. Rest of cells have m=0. Mask bits are never changed.

The mask array is scanned in a row-wise manner and the cell examined -*Current* or *C-cell*- will get the lowest label of three neighbour cells: the *Previous* or *P-cell* at its left, the *Top* or *T-cell* above it, and the *Delay* or *D-cell* at the left of T-cell and above the P-cell. If none of the neighbour cells have m=1, the C-cell having m=1 will get its label from a MN-counter. No labelling is done if the C-cell label is smaller than neighbour labels or if its m=0. MN-counter is incremented after assigning its content to cells without neighbours. Basic labelling rules are shown in Table I.

Table I . Current cell labelling

Top	Prev	Delay	Current cell label
0	0	0	MN - counter
0	0	1	Delay
0	1	0	Previous
0	1	1	Previous
1	0	0	Top
1	0	1	Top
1	1	X	The lowest, Top or Prev

Besides the above rules, it is necessary to consider the cases in which connected cells must be relabelled after detecting inconsistent assignments. These relabelling rules are explained in relation to Fig. 9.

1- The labelling process starts at row0. If no m=1 cells are found, it will jump to row1 and so on. As seen in Fig.9 cells in row1 get MN assignments, starting at 0, the initial value of MN-counter. Cells to the right of cell (1,1) get 0s because of *P-cell* connectivities. Next cells get successive MN-labels after connectivity breaks.

2 - Cell (0,2) gets label 5, the current value of MN after assigning 4 to cell (F,1). However, when reading cell (1,2) and its neighbour cells, a *P>T* state is detected between cells (0,2) and (1,1). A Xflag is set and reverse X-scanning assigns label of cell (1,1) to cell (0,2). During reverse X-scanning, cell (1,1) acts as a *D-cell* to label *C-cell* (0,2), having a higher label. Reverse scanning goes on until reaching column-0 or until a m=0 cell is found. Both cases reset Xflag, restarting forward X-scanning.

3 - To reduce upcounting, MN-counter rised to 6 after assigning 5 to cell (0,2), is decremented to 5 after relabel cell back. To control decrements, a Mflag is set after assigning MN content. Mflag is reset after reading m=0 cells or at column-0. No decrements occur if Mflag=0.

4 - *Top* labelling is done at cell (6,2), acting as *P-cell* to label (7,2), in turn labelling (8,2). When reaching (9,2) a *T>P* state is met, setting a Yflag. This requires relabelling of cell (9,1), but this action will not take place until the full current row is scanned. Further *T>P* states, as in (F,2), will be relabelled by reversing Y-scanning after reaching the current row end. Yflag=1 forces reading of row1, and row2 will act as the source of *T* and *D*- cells.

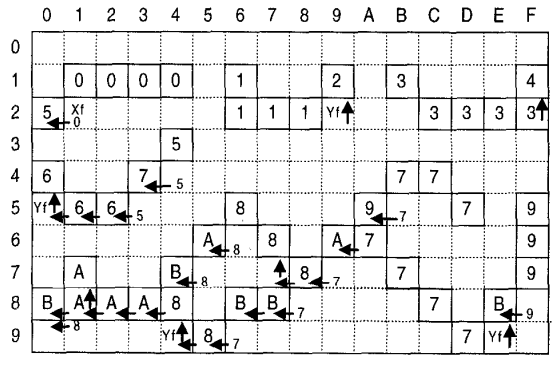


Fig.9 Labelling algorithm example
(m = 1 cells border lined. m= 0 cells border dashed)

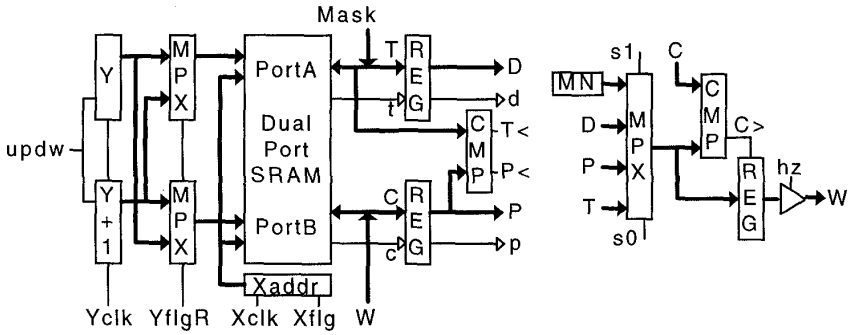


Fig. 10 Implementation of Region Labelling Module

Thus, cell (9,1) will get 1 from its *D-cell* (8,2), and (F,1) will get 3 from its *T-cell* (F,2).

5 - Reverse Y-scanning will keep on until Yflag is reset. This action will happen after scanning row0 or any other row without labellings. This means that a Wflag is set when Yflag=1 and labellings occur. Wflag is tested and reset at row ends. X-reverse scanning is also possible during Y-reverse scanning. Forward Y-scanning will restart on the row below the last one scanned with Yflag=1. In Fig.9, row0 is scanned after labellings in row1. Forward Y-scanning restarts at row1.

6 - Some label numbers can be lost during relabellings. Cell (0,4) gets label 6. However, this label will be lost after relabelling that region. Similarly, labels 8 to B assigned as shown to the complex shaped region at the bottom of Fig.9, will also be lost after relabellings, and the whole region will be labelled 7, the MN-content assigned to the first cell -(B,4)- labelled of that region. Arrows show first relabellings to do, related to Xflag and Yflag settings. If there were m=1 cells at positions (9,9) or (A,9), they would get label B, the last MN value decremented at (E,8).

VI. LABELLER IMPLEMENTATION

It is shown in Fig. 10. To determine connectivity we use a 8K x 8 Dual Port SRAM within the AVD mask module. For the sake of clarity, only main blocks are shown. After loading mask through IO/PortA, scanning starts driven by X, Y and Y+1 address counters. C-cells are at IO/PortB and T-cells at IO/PortA. Previous readings are stored in REG registers, acting as P and D-cells, but their contents are cleared when reading column-0 cells. Signals c, t, p, d contain mask bits. CMP block compares T and P labels, driving T< and P< lines to activate Xflag and Yflag signals registered in D-FFs with additional glue-logic not shown. It can be noticed that registered Yflag swap the Y-counters addresses through the MPXs multiplexers, forcing to appear at IO/PortB a C-cell corresponding to rowY, while at IO/PortA is a T-cell from rowY+1, i.e. Y-reverse scan. Xflag and updw lines set address counters in down mode for reverse scannings. Labelling is done with W signal, the output of MPX receiving the four label sources. Selection of source is made with s0 and s1 signals, functions of the four mask bits c, t, p, d. From Table I, it can be obtained a Karnaugh map to optimize selection functions as follows:

$$MSB \quad s1 = t + p\,; \qquad LSB \quad s0 = d.NOTp + t.T< + t.NOTp \qquad (4)$$

MPX output is compared with label of C-cell. If C-cell label is higher and if c=1, selected source is stored at REG and high-Z output will be removed for memory writing.

VII. BOUNDING-BOX EXTRACTOR

Once the ROIs are labelled, it is possible to obtain their bounding boxes, looking for the X_{min}, X_{max}, Y_{min}, Y_{max} sets with the same label, merely reading the DualPort SRAM.

1. The Bounding-Box marking algorithm.

The procedure is applied as depicted in Fig. 11. The SRAM is scanned row-wise until a m=1 cell is found. Its label is read and stored, as well as its Y-address, i.e Y_{min}. The scan mode is switched to column-wise, from Y_{min} up to finding a m=1 cell with the same label previously stored. When such a cell is found, the Y-address is stored and the X-address stored and incremented. Column scan is repeated until a whole column scan is made without finding the label searched. X_{max} is then the last X-address stored, and together with the still stored Y_{min} determine the BB1 corner of the bounding-box. Its coordinates are ready to be sent to the host, from the registers storing them.

Reverse row scans start at the Y and X-addresses of last m=1 cell found, and repeated when ROI cells are found, after storing addresses. If column0 is reached without hitting ROI cells, the last Y-address stored is Y_{max}. Reverse column scans start at Y_{max}, decrementing X-address after hitting ROI cells, up to reaching Y_{min} without ROI cells. Last X-address stored is X_{min}. BB2 corner can be sent to host from the registers where X_{min} and Y_{max} are now stored.

With the labelling process used, the first ROI found is the 0-label one. Next ROIs will be searched starting row scans at the last Y_{min} and looking for incremented labels.

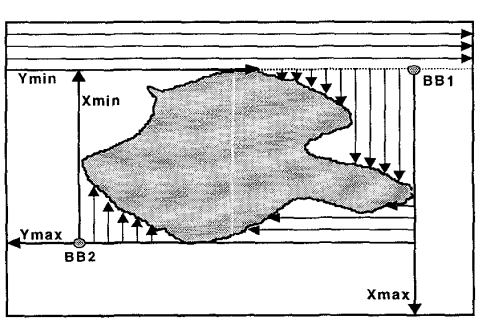


Fig. 11 The bounding-box marking algorithm

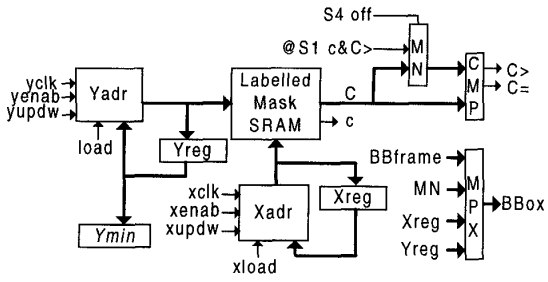


Fig.12 Implementation of Bounding Box Extractor

2. The Bounding Box Extractor Implementation

It is shown at block level in Fig. 12 and follows the marking algorithm described above. It is controlled by a four states Finite State Machine -FSM- and glue logic not shown for the sake of clarity. The states and their order correspond to the scanning modes used in the algorithm. The inputs to the FSM are the c mask bit and four signals -X_0, X_H, Y_{min}, Y_V- to switch the states. The outputs of the FSM are signals iclk, ienab, iload and iupdw, controlling the addressing modes to the SRAM where the ROIs were labelled. Registers, Xreg-Yreg, store the BB1 and BB2 corners and a Y_{min} register is used to store Y_{min} every time a new ROI is hit for the first time. Thus, it is not required to scan the full SRAM to look for the successive ROIs. MN counter is reset at start, since there will always be a 0-ROI. After finishing a FSM cycle -S4off- MN is incremented and gets ready to identify new ROIs. However, if the label of the expected ROI was lost during the application of relabellings, a C> signal will update the MN content. This event is only allowed during state S1. The C= signal validates hits on boundary cells of the ROI stored at MN, skipping over hits on other ROIs.

The interface with host and BBox corners transmission are strongly application dependent. The MPX block shown in Fig. 12 indicates that BB1 and BB2 are sent to host after a BBframe byte -all 1s- followed by the MN content.

Once the host gets the BB corners, according to the high level vision functions being processed, it will instruct the imager for gaze control of foveas and FOV windows thruough the Instruction Bus shown in Fig. 6.

VIII. RESULTS AND CONCLUSIONS

We described an adaptive vision subsystem synthesized into FPGAs -Altera's FLEX 81188- which reduces the burden of operations to be performed by the vision system host. The full hardware implementation of the Bounding Box marking and corner extraction algorithms and the fast convergence for typical convex ROIs, ensure appropriate performance of the tasks assumed at the sensors level, making them suitable for motion applications where speed, system size and cost are relevant factors to consider.

Current clock frequency is 40 MHz. SRAMs used have access times of 25 ns. The number of mobiles and their size in relation to the scenes size, i.e. the data volume handled for levels transmission to host, are the main limits to the performance. Four mobiles occupying around 40% of the scene, working with full 640x480 pixel images are easily processed. Upgrading alternatives have been implemented by dividing the double buffering SRAMs into four blocks to reduce the number of accesses for pixel averaging, reducing fourfold the time of this task.

The number of mobiles or ROIs to be processed can be augmented by increasing the FIFO depth at the Instruction Bus. But it is also a matter of host capability in relation to the high level vision functions being performed.

Usage of FPGA resources is summarized in Table II.

Table II - FPGA Resource Usage

FPGA	IMAGER
Device type	FLEX 81188
Input pins	33
Output pins	74
Bidirectional Pins	32
Reserved Pins	2
Dedicated Inputs	4/4 (100%)
I/O pins used	137/144 (95%)
Logic Cells used	912/1008 (91%)
Flip-Flops used	208

IX. ACKNOWLEDGMENT

This work has been partially granted by Comisión Interministerial de Ciencia y Tecnología, CICYT, under project TIC95 - 0589.

X . REFERENCES

[1] J.Van der Spiegel, G.Kreider, C.Claeis, I.Debuschere, G.Sandini, F. Fantini, G.Soncini, "A foveated retina-like sensor using CCD technology" in *Analog VLSI implementation of Neural Systems*, C.Mead, M.Ismail, Eds., Kluwer Acad. Publ, The Netherlands, 1989.

[2] M.Tistarelli, G.Sandini, "Estimation of depth from motion using an anthropomorphic visual sensor", *Image and Vision Computing*, vol.8, No. 4, 1990, pp. 271-278.

[3] A.S.Rojer, E.L.Schwartz, "Design considerations for a space-variant visual sensor with complex logarithmic geometry", in *Proceedings of the 10th. Internatnl. Conference on Pattern Recognition*, 1990, pp. 278-285.

[4] C.Bandera, P.Scott, "Foveal Machine Vision Systems" in *IEEE International Conference on Systems, Man and Cybernetics*, Cambridge, MA, 1989, pp. 596-599.

[5] E. Schwartz, D. Greve, G.Bonmassar, "Space-variant Active Vision: Definition, Overview and Examples". *Neural Networks*, Vol.8, No. 7/8, 1995, pp.1297-1308.

[6] P.Camacho, F.Arrebola, F.Sandoval, "Adaptive Fovea Structures for Space-variant Sensors", in *Proceedings of the 9th Intnal. Conference on Image Analysis and Processing*, Vol. I, Florence, Italy, 1997, pp. 422-429.

[7] P.Camacho, F.Arrebola, F.Sandoval, "Shifted Fovea Multiresolution Geometries" in *Proceedings of the IEEE Intnal. Conference on Image Processing*, Vol. I, Lausanne, Switzerland, 1996, pp. 307-310.

[8] P.J. Burt, "Smart Sensing within a Pyramid Vision Machine", *Proceedings of the IEEE*, Vol. 76, No.8, August 1988, pp. 1006- 1015.

[9] J.M. Jolion, A. Rosenfeld, *A Pyramid Framework for Early Vision*, Kluwer Academic Publishers, The Netherlands, 1994.

[10] I. Pitas, *Digital Image Processing Algorithms*, Prentice Hall International, UK , 1993.