Multifoveal Imager for Stereo Applications

P. Camacho,* F. Coslado, M. González, F. Sandoval

Departamento de Tecnología Electrónica, E.T.S. Ingeniería de Telecommunicación, Universidad de Málaga, Campus de Teatinos 29071, Málaga, Spain. E-mail: pelegrim@dte.uma.es

Received 15 March 2002; accepted 12 July 2002

ABSTRACT: A multiresolution imager based on adaptive retinal structures, with data compressions above 85%, is presented in this article. The main goal of the imager platform is to speed up image processing with the use of selective data reductions to shorten the vision systems' tasks in stereo applications. Implemented on a field-programmable gate array, the platform can be configured as the front end of active vision systems, with use of adaptive foveal sensing on unireresolution images to cover wide fields of view, being also a development tool for multiresolution applications with different image formats and interfaces. The multifoveal imager provides the hierarchical data structures related to multiresolution levels, following instructions to control sensor parameters or to perform adaptive fovea fixations in real time, adapting its operation to the constraints of the active vision systems. It also uses intermediate resolution data to implement in hardware an efficient background extractor to cooperate with image processors in motion detection tasks and attention mechanisms. Some platform configurations are explained and experimental results are discussed in relation to the advantages of the adaptive retinal structures. © 2002 Wiley Periodicals, Inc. Int J Imaging Syst Technol, 12, 149–165, 2002; Published online in Wiley InterScience (www.interscience.wiley.com). DOI 10.1002/ima.10023

I. INTRODUCTION

Recent technologic advances in the field of CMOS-active pixel sensors have brought to the market tiny megapixel image devices with performances rivaling those of charge-coupled device (CCD) sensors, lower cost, reduced power consumption, versatile interfaces, and data coding for image processing. The current trend in CMOS-device technologies, going below 0.2 μm, will surely reduce pixel sizes; therefore, even larger image formats will soon be available in CMOS sensors, and many new possibilities will arise in some fields of computer vision. However, such high-resolution devices pose new problems related to the required processing speed, storage space to save the images, and channel bandwidths to transmit them.

One of the most challenging areas of computer vision is the so-called active vision (Aloimonos et al., 1988; Bajcsy, 1988) trying to emulate the vision functions and some of the purposive responses observed in the animates. Tasks such as scanning, exploring, and searching are considered basic elements in the process of visual perception and, together with the tasks of deciding where to look next (Swain and Stricker, 1993), encompass a set of relevant functions to implement in active vision systems. Many of the applications related to this field of computer vision require a wide field of view, a matter of looking for some of the many appropriate lenses available; high resolution, something that CMOS devices are already offering; and data processing speed to meet the requirements for system performance. The trade-off among these three concepts has been a topic of interest since the beginning of computer vision; additionally, if systems are considered from an engineering point of view, new constraints arise as to what might be considered a feasible or even a practical system: power consumption, weight, cost, and so on. Thus, the huge data volumes associated with images of high resolution and the wide field of view make difficult the task to achieve acceptable speed of response in stereo vision systems and to apply the algorithms required for segmentation, detection, tracking, and so on, which are essential parts in applications related to dynamic environments, where speed is a must. One of the alternatives to increase speed is data reduction. Under this concept there are many techniques, but even among those oriented to image processing, some will perform properly in specific applications, being unsuitable for others.

Active vision systems are based on the Selective Vision paradigm oriented to a selective data reduction by processing only regions of interest in the tasks under execution, adopting three basic strategies to achieve performances according to scenes contexts: The first is the control of the sensor or camera parameters to adapt their responses to the system requirements; the second is to have the ability to provide attention mechanisms to select the regions of interest; and the third strategy is use of foveal sensing to capture scenes with space-variant resolution to cover wide fields of view, without losing the capability to detect potential events of interest. The Selective Vision paradigm and the active vision systems arose after study and comparison of some of the many different visual systems in the animates, as well as their adaptation to the environment where they live, acting as predators, preys, or both.

The sense of vision in nature is a complex function with two main actions: The act of seeing might be considered a natural, passive, or not a fully intentional function, whereas the act of looking is a much more complex, purposive function related to cognition and implying a brain activity translated into saccadic eye
movements to fix the gaze. The retina, a layer of photoreceptors covering the back of the eyeball, functions to encode the light of the scenes captured through the pupil and transmit it to the brain via the ganglion cells and the optic nerve. In the human eyes, there are almost 100 million photoreceptors in the retina (Panda-Jonas et al., 1994), but only about 1 million ganglion cells are available to carry the information from the photoreceptors to the optic nerve. The approximate 100:1 compression is achieved in a selective manner. About 3 million photoreceptors out of the 100 million are concentrated in a small area of the retina, the fovea or area centralis, covered with a solid angle of about 3° within the almost 130° × 150° vertical and horizontal angular ranges observed in an average retina. The outputs of these photoreceptors, the cones, are connected to about 10% of the ganglion cells. However, the peripheral photoreceptors of the retina, the rods, have only about 900,000 ganglion cells or transmission channels through the optic nerve to convey the information of about 100 million rods, which implies a spatial information compression close to 100:1. The fovea, covering roughly 0.05% of the visual field, takes about 10% of the channels available in the optic nerve, allowing high-acuity vision in the fraction of scenes projected onto it. On the other hand, the density of rods decreases with the distance to the fovea, determining a kind of elliptical ring with decreasing radial resolution.

Vertebrates have adapted their foveae to the environments where they live. Thus, although the most common case is to have a single fovea, there are also animals with two foveae per retina. The shapes and arrangements of the foveae within the retinae differ widely, going from vertical, more or less elliptical streaks, to horizontal streaks, as a result of the animals’ adaptation to some routine functions in their habitats and lifestyles. The importance of foveal shape is worthy of mention, because it helps to reduce the head and eye movements, or saccades, required to look at objects out of the foveal visual field or not completely covered with its small solid angle. Reported studies (Zuber, 1981) have shown high foveation activity even in scenes in which, supposedly, there is a low level of attention. Foveations imply field of view and background changes, a problematic task faced in many artificial vision systems.

The observation and study of biologic visual systems has been a source of ideas for research oriented to emulate them in the fields of computer vision. In multisresolution vision systems, the emulation somehow follows the paradigm of two parallel vision functions, expressed in the sentence, “While the fovea looks, the retina sees.” Different approaches have been taken in space-variant resolution techniques at the sensor level, with log-polar (Van der Spiegel et al., 1989), uniform hexagonal (Mead, 1989), or exponential-Cartesian retinotopologies (Bandera and Scott, 1989). Alternative techniques at the optics level were also proposed (Kathman and Johnson, 1992). Based on the selective data reduction achieved with the imager, our approach is oriented to applications requiring fast processing of stereo images, using hierarchical data structures, corresponding to retinotopologies with adaptive characteristics (Camacho et al., 1996, 1997), to emulate retinæ found in nature with more than one fovea, and to profit the many algorithms developed for rectangular images, unsuitable for non-Cartesian images or requiring complex transformations to adapt them. The algorithms and techniques used at the systems level are beyond the scope of the article.

The subsystem reported in this article is intended as a front end of active stereo vision systems, extracting the hierarchical data structures from uniresolution images supplied by off-the-shelf CMOS sensors to achieve selective data reduction based on multiresolution. In Section II, we present the concepts related to the adaptive retinal structures (ARSs) and the data compressions that could be achieved with them. Section III describes the ARSs and their relations to vision pyramids, presenting the hierarchical data structures used at the systems level and the modified data compression that the imager obtains for these data structures. In Section IV, we refer to the implementation of the stereo imager modules. Finally, in Sections V and VI, we present some of the experimental results obtained with the stereo imager platform and draw some conclusions related to future work.

II. ADAPTIVE RETINAL STRUCTURES

To define the configuration of ARSs, equivalent to description of the retinotopology of an emulated biologic system, we set up a region of interest (ROI) within the field of view (FOV) enclosed in a bounding box determined by the coordinates (Xmin, Ymin) and (Xmax, Ymax) of two opposite corners. The bounding-box dimensions are normally the output of some vision function or algorithm used at the application level. The corresponding retinal structure must contain the full bounding box within its highest resolution region or fovea, emulating the gaze actions performed by the animate visual systems. Surrounding the fovea is a set of m rings whose radial resolution will decrease, emulating the density of peripheral photoreceptors in the retina (Panda-Jonas et al., 1994; Shipley and Shore, 1990).

In our model of ARSs, we set up resolution gradients at each side of the fovea by fixing within all the m rings a set of subrings whose number is constant, but not necessarily equal, at each side of the fovea. The number of subrings is specified with four parameters l, r, t, and b, standing for the left, right, top, and bottom sides of the fovea and, depending on their values, besides determining the resolution gradients at each side in a consistent manner, they force fovea dimensions and position. Thus, in the ARSs in Fig. 1(a and b),

![Figure 1](image-url)
corresponding to a 160 × 120 pixels image, there are three resolution rings ($m = 3$) surrounding a fovea whose size and position within the FOV become determined by the width of the rings (i.e., the number of subrings $l$, $r$, $t$, and $b$ at each side). To emulate the decreasing density of rods in the periphery of the retina, the receptive field (i.e., the size of the maxels or superpixels contained in the rings) increases from the fovea to the periphery of the FOV following a $2^N$ law, fixing the resolution ratio between successive resolution regions or rings. Thus, the size of maxels or pseudorods in the first ring surrounding the fovea is $2 \times 2$ pixels; that is, their resolution is one fourth of the fovea pixels or pseudocones, and the outer ring resolution is one fourth of that in the middle ring. Considering that the outward linear resolution ratio of rings is $2$ in any direction, the common dimension factor in the ARS with $m$ rings is

$$J_N = 2^m + 2^{m-1} \ldots + 2^{N+1} \sum_{N}^{m} 2^{N-1} (0 \leq N \leq m - 1),$$

(1)

where $J_0$ represents the minimum jump, in pixels, to shift the fovea or to resize its dimensions. When considered for mathematical reasons, $J_m = 0$ means that the outer ring remains fixed, because it is at the periphery of the FOV. According to Eq. (1), placing the origin of coordinates at the upper-left corner of the structure and choosing a convenient low-resolution limit (i.e., the number of rings $m$), $J_0$ becomes fixed, and the smallest fovea able to contain the bounding box specified by $(X_{\text{min}}, Y_{\text{min}})$ and $(X_{\text{max}}, Y_{\text{max}})$ has dimensions and position determined by the following expressions for the ring subdivision factors:

Left factor $l = \left[ \frac{X_{\text{min}}}{J_0} \right]$ \hspace{1cm} Right factor $r = \left[ \frac{H - X_{\text{max}}}{J_0} \right]$

Top factor $t = \left[ \frac{Y_{\text{min}}}{J_0} \right]$ \hspace{1cm} Bottom factor $b = \left[ \frac{V - Y_{\text{max}}}{J_0} \right].$

(2)

After obtaining the factors, the coordinates of the two opposite corners of either the fovea or any of the internal rings are given, in pixels, by

$$X_{\text{max}} = J_0 l \hspace{1cm} Y_{\text{max}} = J_0 t \hspace{1cm} X_{\text{max}} = (H - J_0 r) \hspace{1cm} Y_{\text{max}} = (V - J_0 b),$$

(3)

meaning that both the eccentricity and the size of foveae and rings can be fixed by changing appropriately the value of the factors. It allows the emulation of retinotopologies adapted to the size and position of the objects of interest in the FOV, reducing to just one the number of foveations or saccades required to observe fully or look at them. Retinal images can be thought as a transform of the uniresolution images taken with uniform sensor arrays, showing the elements that an electronic retina would send to the vision system processor. The retinotopologies are not limited to foveae fully surrounded with a set of rings. Foveae could also be located at the periphery or at the corners of the FOV, if one or two subdivision factors are set to zero. Several foveae can be defined in an ARS, obtaining multifoveal structures such as the bifoveal one shown in Fig. 1(c), after merging Figs. 1(a and b). The four arrows in Fig. 1(c) are related to vision pyramids, and we comment on their meaning in Section III. From Eq. (3), we can be derived the data compression obtained with unifoveal ARSs. Considering the size of the elements in foveae and rings, their outer lateral dimensions become

Fovea or ring width $w_N = \frac{1}{2^N} [H - (l + r)J_0] = \frac{1}{2^N} [H - x_{J_0}]$

Fovea or ring height $h_N = \frac{1}{2^N} [V - (t + b)J_0] = \frac{1}{2^N} [V - y_{J_0}].$

(4)

and based on Eq. (4), we can obtain the number of elements within the fovea and the rings. Thus, the number of pixels in the fovea, considered as ring 0, is

$$E_{r,0} = E_0 = w_0 h_0 = [H - x_{J_0}][V - y_{J_0}],$$

(5)

but in the rings, to obtain the number of maxels, we subtract the areas defined within the inner subrings from those defined by the outer subrings, both given in maxels of the considered ring:

$$E_{r,N} = \frac{1}{2^N} [H - x_{J_0}][V - y_{J_0}] - \frac{1}{2^N} \times [H - x_{J_{N-1}}][V - y_{J_{N-1}}],$$

(6)

which can be rewritten as

$$E_{r,N} = E_N - \frac{1}{4} E_{N-1},$$

(7)

and applying, recursively, Eq. (7), the total number of elements in the ARS becomes

$$E_T = \sum_{0}^{m} E_{r,N} = E_0 + \left( E_1 - \frac{1}{4} E_0 \right) + \cdots + \left( E_m - \frac{1}{4} E_{m-1} \right) = E_m + \frac{3}{4} \sum_{0}^{m-1} E_N.$$

(8)

Applying Eq. (4) into Eq. (8), and considering that $J_m = 0$, after some mathematics, we obtain

$$E_T = HV - \frac{3}{4} \sum_{0}^{m-1} J_N \left[ H_0 + Vx - xy_{J_N} \right],$$

(9)

which can also be rewritten as a function of the fovea dimensions $w_0, h_0$ obtained from Eq. (5), instead of $x, y$:
Defining the retinal data compression (RDC) in relation to the original dimension of the FOV, we obtain the figure of merit of the ARS as

\[ E_T = \frac{HV - 3}{4J_0} \sum_{i=0}^{m-1} \frac{J_N}{2^m} \times \left[ 2HV - HH_0 - VW_0 - \frac{J_N}{J_0} (H - w_0) (V - h_0) \right]. \]  

(10)

Thus, in the ARS of Fig. 1(b) we obtain \( E_T = 1812 \), giving an RDC of 90%.

As might be expected, the bigger the fovea, the lower the number of elements in the rings and the lower the data compression of the ARS. Fig. 2 is a three-dimensional (3D)-plot of RDC for \( m = 3 \) and \( H \times V = 640 \times 480 \) pixels, showing that RDC is in the range of 97.0 to 88.5% for foveae up to \( 164 \times 144 \) pixels, which should be considered excessive for multiresolution applications working on VGA images. Multiresolution would not make much sense if the sizes of foveae force poor data compression.

### III. ADAPTIVE RETINAL IMAGES AND VISION PYRAMIDS

The concept of vision pyramids (Tanimoto and Pavlidis, 1975) is a well-known subject in the field of multiresolution and a wide research area in the field of computer vision (Jolion and Rosenfeld, 1994; Rosenfeld, 1984). Although pyramidal levels could be easily obtained by subsampling, this method would conduct to a rather high frequency component, or noise, as the level resolution decreases, introducing aliasing into the lower resolution images. As a consequence, many of the algorithms applied at systems level for hierarchical segmentation (Burt et al., 1981; Cibulskis and Dryer, 1984; Hong et al., 1982) and block matching (Jong et al., 1994; Nam et al., 1995) could require a long time to converge. Hence, to maintain our goal of fast systems responses, we construct image pyramids by using lowpass filtering. The lowpass filter choice is a trade-off, too: Gaussian lowpass filters reduce the noise and smooth image compression artifacts, but they imply heavy computational complexity, without presenting clear advantages in processing speed at the time of applying high-level algorithms. Therefore, pyramid levels in our work are constructed by successive 4-to-1 averaging:

\[ g_L(i, j) = \left[ \frac{1}{4} \sum_{u=0}^{1} \sum_{v=0}^{1} g_{L-1}(2i + u, 2j + v) \right] \quad (0 \leq L \leq 3), \]  

(12)

where \( g_L(i, j) \) is the gray level of the maxel \((i, j)\) of the \(L\)th level, and \( g_0(i, j) \) represents the pixels of the original image or pyramid base. Although we have limited the number of levels to 4, this figure could be higher depending on the optics angle, the sensor resolution, and the maximum size of targets we expect to fix in the fovea. In fact, the higher the \( m \), the higher the data compression, as derived from Eq. (11). In our work with VGA images and wide-angle optics, \( L \geq 3 \) or \( m \geq 3 \) is useless because of the high loss of resolution implied at the periphery of the retinal image, compressing \( 16 \times 16 \) pixels into one maxel. However, some of the hierarchical segmen-
tation algorithms mentioned need \( L = 6 \), or more, in order to fix a moderate number of segmentation classes and to maintain an acceptable time for convergence, but the reduced data volumes associated with levels beyond \( L_3 \) do not significantly penalize the computational load left to the processor to generate them, in the case of using those algorithms.

A. Adaptive Retinal Structures in Relation to Vision Pyramids. The relation between ARSs and vision pyramids can be determined by observing Figs. 1(c) and 3. For the sake of clarity, in Figure 1(c), showing an adaptive bifoveal structure, four lines have been drawn connecting the corners of only one of the two foveae to the corners of the FOV. Although the fovea must always be considered a fraction of pyramid base, \( L_0 \), the boundaries of the ARSs are normally related to the outer ring (i.e., the one with the coarsest resolution and, therefore, related to the upper level, \( L_3 \)) of the pyramid in Fig. 3. The exception is those structures having a fovea located at the border of the FOV. The size of the maxels in the peripheral ring indicates that their gray values are the average of all the pixels they encompass if the ARS is mapped onto the FOV, but it also means that the peripheral maxels correspond to the lowest resolution level of the associated pyramid. In fact, the number of horizontal and vertical maxels on the outer subring in Figure 1(c), 20 and 15, respectively, correspond to the \( H \times V \) dimensions of level \( L_3 \) obtained after applying three successive averagings on the 160 \( \times \) 120 pixels of the pyramid base image or FOV. Observing the pyramid in Fig. 3, it can be seen that the four arrows going from the fovea corners up to the corners of the top level \( L_3 \) correspond in a 3D-space to the four arrows in Figure 1(c). Those arrows intersect the intermediate levels \( L_1 \) and \( L_2 \), determining in them ROIs whose size and position are related to the size and eccentricity of the corresponding fovea within the FOV. The ratio of any ROI size to the size or dimensions of its level increases as the resolution of the level decreases (i.e., the solid angle of ROI at \( L_2 \) is wider than the solid angle of ROI at \( L_1 \)).

The fovea corresponds, or should correspond, to the minimum solid angle required to cover with high resolution the target of interest in the FOV. Level \( L_3 \) should have the minimum resolution needed to observe the full FOV, keeping the ability to detect potential details of interest on it at such a level of low resolution. The lines connecting the corners of the ROIs, from fovea to \( L_3 \), form a kind of tilted, truncated pyramid within the full pyramid. The tilting of the lateral faces of the truncated pyramid depends on the position and size of the fovea. If the borders of the full intermediate levels \( L_2 \) and \( L_1 \) are projected onto the pyramid base, so as to form new truncated pyramids with lateral faces parallel to those of the inner truncated pyramid, the projections determine the resolution rings around the fovea and, therefore, the retinal images that the adaptive sensor structures would capture. In spite of the space-variant vision emulation, retinal images do not contain the hierarchical data structures required at systems level, although these could be extracted from the retinal images. Therefore, retinal images are not used at the systems level.

B. Data Compression of the Hierarchical Data Structures. The hierarchical data structures used at system level are the set of ROIs starting at the fovea and ending up at the level \( L_m \), where the full FOV is compressed. They form a set of uniresolution images whose resolution decreases while the solid angle increases. There are at least two alternatives to transmit the multiresolution data from the retinal imager to the systems processor: One is the retinal image, with the advantage of being the most reduced data set that could be transmitted, whose compression was given in Eq. (11), and the disadvantage of the computational load left to the processor to extract the ROIs from the retinal images. The second alternative is to transmit the ROIs from the imager, implying a redundant data
sending (e.g., the ROI of the first level contains the first ring of the retinal image and fovea data at the resolution of level $L_1$). To evaluate the data compression achieved with this alternative, we proceed in a manner similar as that done to obtain Eq. (10) but, in this case, applying the same symbols used in Section II, the total number of elements sent to the processor is

$$E_T = E_0 + E_1 + \cdots + E_m = \sum_{n=0}^{m} E_N$$

$$= \sum_{n=0}^{m} \frac{1}{2^n} [H - xJ_0][V - yJ_3], \quad (13)$$

which, considering that $J_m = 0$, can be rewritten as

$$E_T = \sum_{n=0}^{m} H V \frac{1}{2^n} - \sum_{n=0}^{m-1} \frac{1}{2^n} [(H y + V x)J_0 - xyJ_3], \quad (14)$$

where the first element at the right side represents the number of maxels within the pyramid levels, and the second element represents the number of maxels within the rings surrounding the ROIs at pyramid levels below $L_m$. Therefore, with the same definition for data compression used in Eq. (11), applying to Eq. (14) the same substitutions in Eq. (9), the new figure of merit, or pyramidal data compression (PDC), becomes

$$PDC = \frac{1}{HVJ_0} \sum_{n=0}^{m-1} J_n \frac{2^n}{2^n}$$

$$\times \left[ 2HV - Hh_0 - Vw_0 \right] J_0 (H - w_0) (V - h_0) = \sum_{i=1}^{\infty} \frac{1}{2^n}. \quad (15)$$

As a comparison with the RDC achieved with Eq. (11), applying Eq. (13) yields $E_T = 2316$ for the same Fig. 1(b), which implies that PDC = 88%. The 3D-plot in Fig. 4 shows that the differences between RDC and PDC are below 3.5% for the same FOV and range of foveae in Fig. 2.

C. Multifoveal Structures. The emulation of retinae with more than one fovea, as shown in Fig. 1(c), is obtained by merging two or more ARSs with a single fovea. The main interest of multifoveal structures resides in the ability to process in parallel the reduced data sets corresponding to targets or events occurring in a single image or frame of a sequence. Depending on the algorithms applied at the systems level, multifoveal data processing may provide information useful in deciding the most relevant target to analyze in a given context. From the vision system point of view, the processing of every ROI will probably be independent from the others; however, depending on the size and proximity of foveae within the FOV, it is possible to have elements at any level, i.e. pixels or maxels, be common to two or more ROIs, as can be seen in second ring of Fig. 1(c), or level $L_2$ in Fig. 3. This fact implies two considerations: The first is the coding of those elements at the imager to enable their association to all the ROIs to which they could pertain. The second is the multifoveal data compression, a complex function related to the number of foveae, their sizes, and relative positions within the FOV. Because the top level $L_m$ is unique and the ROIs may have common elements, the number of elements in a hierarchical multifoveal structure is lower than the sum of elements corresponding to the unifoveal structures merged into it. Therefore, the transmission of multifoveal images has a higher compression, but this does not mean that the RDC or PDC is low, because both figures of merit are related to the total number of elements transmitted to the vision processor (i.e., to the sum of elements of the overall multifoveal structure). Because of the small difference between both data com-

Figure 4. Compression loss of the pyramidal data structures.
expressions, shown in Fig. 4, adopting a systems engineering strategy, we choose the alternative of transmitting the pyramidal ROIs, instead of retinal data, to relieve the processor of routine tasks that can be done easily and faster at imager level. In this case, it is necessary to determine the coordinates of all ROIs within their respective pyramid levels, in a similar manner as that done in Eq. (3) to obtain the coordinates of any ROI at any level, related to the origin corners at those levels, are given in their FOV where they are mapped. The coordinates of any ROI at any level, related to the origin corners for the rings within the ARS, related to the pyramids levels, in a similar manner as that done in Eq. (3) to obtain  

\[ \text{ROI coordinates at the pyramidal levels minimizing hardware resources.} \]

IV. IMAGER IMPLEMENTATION

A block diagram of the stereo imager is shown in Fig. 5, representing the main common blocks and the left channel blocks. The platform operates under instructions issued through the applications running on a 1GHz PC. Instructions to CMOS sensors are intended to optimize their responses under different illumination conditions, fixing the light integration time, the gain, and the ADC reference levels. Sensors are initialized at application setup and, if necessary, their appropriate registers modified via the two I2C interface signals, sclk and sdat. At the design-specification phase of the stereo platform, we had in mind its intended use as a tool to develop multi-resolution applications whose specifications concerning speed and data bandwidth may differ widely, so we have incorporated two systems interfaces into the platform; one to the parallel port of a PC, and the other to a digital acquisition board with 120 MB/s bandwidth, connected to the PCI-bus. Fovea-fixing instructions could have two different formats, the first being by reception of the \( l, r, t \) and \( b \) parameters, which requires 3-byte-wide instructions, and the second using the bounding-box coordinates \((X_{\text{min}}, Y_{\text{min}})\) and \((X_{\text{max}}, Y_{\text{max}})\), which would require 5-byte-wide instructions. Considering the simplicity of operations with sums of powers of 2, as implied in expressions given in Sections II and III, both the applications and the Instructions Decoder use the \( l, r, t, b \) format. The Coordinates Generator, applying recursively Eq. (18), generates the coordinates required in the ROI Extractor, whose structure and operation are closely related to those of the Pyramid Generator. External 32K \( \times \) 9 bits FIFOs are used to implement an adaptive algorithm for background extraction and updating, as a reference for image differencing and mobiles detection at the low resolution of level L2. After this key function, the bounding boxes of moving regions are easily obtained at systems level and converted to \( l, r, t, b \) parameters to define the foveae sizes and positions. Because the

\[ \begin{align*}
X_{m(N)} &= \frac{1}{2^N} (J_N t) \\
Y_{m(N)} &= \frac{1}{2^N} (J_N t) \\
X_{M(N)} &= \frac{1}{2^N} (H - J_N r) \\
Y_{M(N)} &= \frac{1}{2^N} (V - J_N b),
\end{align*} \]

where \( m(N) \) and \( M(N) \) stand for minimum and maximum of the respective coordinates at the \( N \)th level. However, the above expressions are not convenient for implementation in FPGAs because of the resources that would be required to implement multiplications; therefore, considering the expression given in Eq. (1) for \( J_N \) and the associated recursivity

\[ J_{N+1} = J_N - 2^{N+1}, \]

the coordinates in Eq. (16), after some substitutions, can be rewritten as

\[ \begin{align*}
X_{m(N+1)} &= \frac{X_{m(N)}}{2} - l \\
Y_{m(N+1)} &= \frac{Y_{m(N)}}{2} - t \\
X_{M(N+1)} &= \frac{X_{M(N)}}{2} + r \\
Y_{M(N+1)} &= \frac{Y_{M(N)}}{2} + b.
\end{align*} \]

Thus, after determining \( l, r, t, \) and \( b \), fovea coordinates are obtained with Eq. (3) and, applying Eq. (18) recursively, we can get ROI coordinates at the pyramidal levels minimizing hardware resources.

Figure 5. Block diagram of left channel and common modules of the stereo imager platform.
A. The Pyramid Generator. The Pyramid Generator consists of a snake-pipeline that recursively processes pixels entering from the unitresolution sensors. The Maxels Generator block in Fig. 6 has three pipelines. The first one processes pairs of adjacent image lines, adding two consecutive pixels in a first adder and, when lines are odd, storing the sums in a FIFO. FIFO sums are read during even lines and added with sums of corresponding pairs of pixels in these lines. The sums from the second adder, right-shifted two positions, are the gray levels of maxels pertaining to the $L_1$ pyramid level, implementing the first level of Eq. (12). Simultaneously, pairs of generated $L_1$ maxels are added in the second pipeline, in a process identical to the one described above for pixels entering the first pipeline. The results, the maxels of $L_2$ pyramid level, enter the third pipeline to produce at its output the maxels of the $L_3$ pyramid level, completing the implementation of Eq. (12). A set of counters keeps the current coordinates of pixels entering from the sensor, $L_0$, and maxels generated for levels up to $L_{m-1}$.

The total delay between the last pixel of a frame and the generation of the last maxel is below 400 ns (i.e., all pyramid levels are practically obtained in coincidence with the entrance of pixels from the sensors). Fig. 7 depicts maxel signals timing in relation to pixels, showing that there is also a $2^n$ ratio among clock rates, whose duty cycles, and interleaving allow the addition of as many levels as desired, a convenient feature when using megapixels sensors. As a consequence of the parallel generation, some storage device is needed to act as a buffer between the pyramid generator and the processor interface. A FIFO alternative was chosen, using the SRAM resources of the FPGA.

B. The Regions of Interest Extractor. The function of the ROI extractor block in Fig. 6 is the extraction of fovea pixels and ROI maxels corresponding to the hierarchical data structure explained in Section III. ROI extractor operation starts loading the coordinates of the regions of interest into the Corners Register. Although fovea coordinates are sent by the vision processor at any time, coordinates loading occurs only during the vertical blanking periods or interframes times, ensuring that only full frames are processed and full sets of ROIs are extracted. The coordinates of pyramid elements, registered in $L_i$ counters, are compared with the Corners registers contents and, if counts are within the range of ROI coordinates, time-window signals are sent to enable maxels writing into the FIFO, buffering the outputs from the pyramid generator. Because the whole top level $L_m$ is always sent to the processor, $L_3$ maxels do not enter the ROI extractor, going directly to the FIFO. In contrast with the process followed with maxels, fovea pixels are directly sent to the processor through the interface multiplexer. At this point, it is worth noting that, as can be derived from Fig. 3, the number of maxels depends on foveal dimensions, whereas the progressive scan determines that the time at which they are generated depends on foveal positions. This means that the sequence of maxels entering the FIFO is a rather complex function depending on both the fovea dimensions and position. Although maxels generation at any level are time-ordered, they enter the FIFO in an interleaved manner, as seen in Fig. 7. The sequence may start at $L_1$ or $L_3$, continuing on for a certain number of maxels, and afterwards, alternating the level or $L_i$ register from which the maxels are caught. To overcome the complexity of the sequence, all maxels entering the FIFO are labeled with 2-bit headers identifying their levels. For the same reason, fovea pixels are headed with "00." This strategy makes quite easy pixels and maxels filtering to reconstruct the ROIs at

![Figure 6. Block diagram of pyramid generator, regions extractor, and imager interface.](image)

![Figure 7. Pixels and maxels interleaving at the pyramid generator output.](image)
specific buffers created in the processor for each level or header. Concerning the FIFO reading, we must refer to Fig. 8, describing the sensors readout timing. As can be seen, there are long horizontal blanking times after each image line or row. During blanking times of even lines, the ones originating maxels, the FIFO is read and emptied. The maximum number of maxels to store in the FIFO is related to the maximum foveal width allowed. Thus, with VGA images and foveae up to 164 pixels, the maximum number of maxels, or FIFO depth, is 288.

Fovea heights have no constraints within the imager implementation, being limited to 144 pixels, or 30% of VGA height, to keep PDC compression above 85%. With similar reasoning for 320 × 256 images, foveal sizes were limited to 82 × 74 pixels, in order to have a similar PDC and to fit within the limited bandwidth of the enhanced parallel port (EPP). The maximum number of maxels that could be generated in rows, multiples of 8, as shown in Fig. 9, is 144 instead of the 288 maxels for VGA images.

Multifoveal structures imply an expansion of the above, each fovea requiring its own ROI extractor. It is worth noting that only one pyramid generator and only one set of counters are required in case of multifoveal structures. However, each set of ROIs has its own set of corners, registers, and coordinates comparators. Whereas headers just identify levels, multifoveal structures increase coding requirements due to the different ROIs present within the same level. An alternative coding method is be explained in subsection D.

C. The Stereo Platform Interfaces. The interface shown in Fig. 6 corresponds to applications using a PCI-bus data acquisition board, customized as a specific frame grabber. During the development of some applications, besides the multiresolution ROIs, it is also convenient to have the full FOV uniresolution images to test the application performance (e.g., processing delays or system responses to input images). This implies headers expansion to code FOV images. Thus, “100” is used for nonfoveal pixels and “000” for foveal ones. Headers field expansion is also used to code additional data, such as background images and mask bit m, discussed in the next section, using headers “11m.” The 32 I/O interface allows parallel transmission of the two channels and level headers mentioned above for bifoveal structures.

Once multiresolution applications are developed, FOV images are no longer needed, and the bandwidth requirements for imager data transmission will probably be supported by simpler PCI-boards or other PC interfaces, such as EPP, in spite of its limited bandwidth. This interface of the platform, shown in Fig. 10, was developed to work with 320 × 256 images, with readouts shown in Fig. 8, supplying alternatively data from both channels at 12.5 fps, or one channel at 25 fps. Foveae limited to 82 × 74 pixels originate upper ROIs of 58 × 50, 46 × 38, and 40 × 32 maxels, as indicated in Fig. 9. To prevent data losses, only two types of elements are sent during a row time. Thus, reading of FIFOs 2 and 3 are delayed 1 and 3 lines, respectively, after the one generating them, as shown with arrows in Fig. 9. The interface multiplexes pixels and maxels, as described in the flow diagram of the Finite State Machine (FSM) in Fig. 11, controlling the transmission. FIFOs are read at 0.75 MHz, and the FSM inserts 4-byte headers before transmitting full FIFO contents to identify their elements. Therefore, the maximum number of bytes transmitted during any row time is 148, including the 8 bytes of two headers formatted FF 00 FF 0L, the L-level code. The 34 vertical blanking lines shown in Fig. 8, or 44 in case of VGA images,
provide interframe times long enough to receive instructions from the system processor.

**D. The Multifoveal Interface.** Multifoveal images, as discussed at the end of subsection IV-B, besides level coding, require ROI coding, with the special case of elements being common to more than one region after merging. These elements must be identified to ease the process of filtering and ROI reconstruction at the processor side. Whereas the multifoveal imager requires as many ROI extractors as expected foveae, every ROI extractor has a flag bit that is activated during the time window corresponding to its elements. In our work, we have limited the foveae to four, so, the ordered ROI flags form a nibble that is used as a header of the pyramid element, to identify the ROIs containing it. The multifoveal interface is an expansion of the interface described in Fig. 10, with four FIFOs, whose widths are increased to 12 bits, to accept the nibble header. During transmission of pixels or maxels contained in the FIFOs, the multifoveal FSM receives the flags header as an additional input.
and, when detecting more than one flag activated, inserts a 4-byte sequence with format FC 00 FC 0R, R, being the nibble found in FIFO header. As soon as the FIFO header changes, a new FC 00 FC 0R string is sent to the processor. Thus, instead of increasing the interface width, 4-byte coding sequences are used to code regions (FC ~ 0R) and levels (FF ~ 0L), keeping the stereo interface width limited to just 1 byte per channel. This technique has proved to be efficient and reliable, without implying any decrease in image-processing performances because of processor dedication to string-header filtering.

Nevertheless, multifeature applications must keep the basic advantages of multiresolution (i.e., data compression and fast processing of the multifeature hierarchical data structures). Although the latter is mainly dependent on the algorithms applied at the systems level, the former affects imager concepts and required FPGA resources. Thus, although foveae are limited to four, data compression depends on regional areas, so the sum of foveal areas is limited to 192 × 186 at application level, to keep compression above 80%. But this is not the only constraint: Horizontal narrow-streak foveae cannot be just any length, because of the potential FIFO overflow during the long, but limited, horizontal blanking times. Thus, in spite of having four multiplexed FIFOs at the interface, the maximum sum of foveae widths is limited to 220 pixels for VGA images.

E. The Background Extractor. Some vision applications, such as those for surveillance, use background differencing related to algorithms for mobiles detection and later extraction of the bounding boxes determining foveae coordinates. Initial background references imply user-supervised tasks dependent on scene contexts, sometimes requiring an average of N images taken in a temporal window of a sequence, but it is also possible to take a single-shot image, if it is considered accurate enough to represent the background that, in any case, will be updated according to illumination changes or to improve the initial reference.

Background-updating techniques consist usually of a progressive substitution of the regions changing their luminance either due to smooth light variations or, eventually, to incorporate potential moving objects entering the FOV and staying quiet after a while, becoming part of the updated background (e.g., the cars entering a parking area).

Our model for background updating is a hardware-adapted version of a widely referred algorithm (Karmann and von Brandt, 1990), consisting of a recursive relation to insert changes into the background at two different speed factors, $F_t$ and $S_t$:

$$B_{ij,t} = B_{ij,t-1} + (F_t - S_t M_{ij,t})(I_{ij,t} - B_{ij,t-1}) = B_{ij,t-1} + (F_t - S_t M_{ij,t})D_{ij,t} \quad (19)$$

where $B_{ij,t}$ represents the gray value of a background pixel at time $t$, $I_{ij,t}$ the corresponding pixel in the current input frame, and $M_{ij,t}$, a binary mask defined in relation to a threshold $T_t$ as

$$M_{ij,t} = 1, \text{ if } |D_{ij,t}| > T_t \quad M_{ij,t} = 0 \text{ otherwise.} \quad (20)$$

Thus, if the absolute value of the difference (AVD) is below $T_t$, it is assumed to be due to light changes; therefore, the background will be modified according to the faster factor $F_t$ after a short number of frames, if the change persists. On the other hand, if AVD is higher than $T_t$, the assumption is that a moving object entered the FOV and is temporarily occluding the background, which will be updated with the slower factor $(F_t - S_t)$, to maintain the AVD longer and to cause object segmentation from background. Criticism arise for both assumptions: Small changes of luminance could be due to moving objects, with poor contrast with the background. In fact, human vision could probably have a similar problem. On the other hand, abrupt changes might be caused by sudden changes such as shadows, reflections, and so on, instead of moving objects, particularly in outdoor scenarios. These problems, normally neglected by vertebrates, pose the hardest constraints at the time of avoiding false alarms; therefore, a longer time is fixed to incorporate them into the background using other algorithms in parallel, such as edge detection, to identify objects in a more robust manner. Because these are computationally expensive tasks, our approach is to solve part of the problem at the imager level, reducing the burden left to the vision processor.

The subsystem operates with four inputs: images $I_{ij,t}$ and systems-generated controls $T_t$, $F_t$, and $S_t$, all three related to scene context and, therefore, subject to be changed by the system at any time to maintain the efficiency at an acceptable level, based on the two outputs—the updated background and the moving mask. Under this scheme, we have implemented the subsystem shown in Fig. 12, where $I_{ij,t}$ are L2 images from the pyramid generator, supplying a reduced data volume, $160 \times 120$ maxels, with a moderate compression and acceptable capacity to detect relatively small moving objects. An additional advantage is that, after two low-pass filters, potential noise in full-resolution images becomes reduced.

Subsystem operation starts loading one L2 image at an instant without moving objects. In case there are any moving objects, user and system have to wait a few seconds in a supervised mode for background stabilization and removal of the vanishing effects caused by the moving objects. At startup, fixing initial values is required for $T_t$, $F_t$, and $S_t$, based on scene luminance and expected speed range of moving objects. Appropriate instructions are issued from the processor, as indicated in Fig. 5, to load their values into internal registers of Comparator and Factor blocks shown in Fig. 12. Normal operation obtains the AVD, or $|D_{ij,t}|$, comparing pixel to pixel, the incoming $I_{ij,t}$ and $B_{ij,t-1}$ stored in the external FIFO. Because of the low rate of L2 maxels, synchronization is not critical. $|D_{ij,t}|$ and $T_t$ are compared to obtain $M_{ij,t}$ bits, following Eq. (20). Depending on the mask bit, $|D_{ij,t}|$ is multiplied by either $F_t$ or $(F_t - S_t)$ and the result added to or subtracted from $B_{ij,t-1}$, as a function of a sign bit generated at the AVD module, obtaining the updated $B_{ij,t}$, as indicated in Eq. (19), which is loaded into the FIFO to serve as a new background during the next frame.

Because factors $F_t$ and $F_t - S_t$ are smaller than 1, their values ranging between 0.25 and 0.04 for $F_t$, and down to one eighth of that range for $(F_t - S_t)$ to speed up the process, the Factors block is actually a divider, implementing into two clocks a fast and simple hardware division algorithm, the divisors being positive integers from 1 to 255, sent from the processor.

As is apparent, FIFO width allows simultaneous storing of $B_{ij,t}$ and the $M_{ij,t}$. Their transmission to the vision processor takes place during the vertical blanking of sensors, configured at imager startup to have 44 rows, as shown in Fig. 8, so it has no relation to the interface FIFO constraints and data related to horizontal blanking. During the interface periods, the pyramid generator and $B_{ij,t}$ generation blocks are inactive, so the 32K × 9 FIFO is then set in a recirculating read/write mode to transmit $B_{ij,t}$ and $M_{ij,t}$ to the processor, reloading them back into the FIFO before the start of the next frame. Transmission of the $160 \times 120$ elements of $B_{ij,t}$ and $M_{ij,t}$ takes less than 0.8 ms, remaining at about 2.5 ms for the
processor to send new instructions to the imager. To identify the \( M_{i,j,t} \), \( B_{i,j,t} \) pairs at the processor, they are labeled "11\( m \)", where \( m \) stands for the \( M_{i,j,t} \) bit value, as shown in Fig. 6.

V. RESULTS

Imager design was implemented in an EPF10K50ETC-144 FPGA from Altera offering 50 K gates and 40 K bits of SRAM. To optimize FPGA resource usage, the design was mainly structural, using graphic design files (.gdf) and customized megafunctions generated with the Max+PlusII development tool from Altera. Roughly 20% of the design was VHDL-based. The progressive-scan monochrome sensors are VGA types PB-0300 and PB-0100 with CIF resolution (352 H \( \times \) 288 V), \( \frac{1}{4} \) and \( \frac{1}{2} \) inch, respectively, CMOS digital image sensors from Photobit, both offering camera-on-a-chip functionality through a set of registers accessible via a simplified I2C interface. The imager power consumption is under 800 mW.

Besides VGA images, we have used 320 \( \times \) 256 pixel images to compare systems operational limits at low and high resolutions, combining image sizes, optics, and PC interfaces. At applications startup, image formats are fixed at the sensor configuration phase, defining windows within their default readouts. Depending on the selected interface, sensors, and image formats with which to work, the FPGA is configured with different EPROMs. Imager instructions are issued to the FPGA to change the fovea or the imager frame rate. Because the sensors are always synchronized and operating at 25 fps, requiring a rather long register-setting procedure to change properly their frame rates, one of the instructions is used to change the imager frame rate by processing 1 out of \( N \) images entering from the sensors. The prototype shown in Fig. 13 uses low-cost 12 M \( \times \) 0.5 mm lenses incorporating IR cutoff filters, with \( F \# \) 2 and focal lengths of 6 and 8 mm, providing FOVs of 60° and 40°, respectively, with optical distortions below 4%. The PCI-bus interface board, with 32 configurable I/O lines, supports well enough the stereo application bandwidth requirements expected from the system using VGA images.

The adaptive retinal images in Fig. 14 are synthesized at application level from the hierarchical data provided by the imager. They show the radial resolution decrease, which becomes more evident in low-format images, as in Fig. 15, representing a pair of 320 \( \times \) 256 stereo images captured with 40° lenses. The retinal compression distortion is less evident in the ARSs obtained from VGA images, as shown in Figs. 16 and 17, with the same number of rings and a 60° angle lens to cover the uniform VGA array. Apparently, the peripheral compression in the ARS from VGA images seems to be like that in the second ring of the 320 \( \times \) 256 image, because of the maxel sizes in relation to the FOV.

The retinal images show clearly at their peripheral rings the low resolution of the upper pyramid level, hardly appreciated in Fig. 15. Retinal images are merely useful to estimate the system capability for detections at the peripheral rings in relation to scenes context, distances, and objects sizes. This is because of the emulation of the retinotopology in human vision, but they are not used at the systems level because of the complexity associated to their space-variant distortion. Thus, in Fig. 14, comparing the image size of the person within the fovea with the size of peripheral maxels, compressing 8 \( \times \) 8 pixels, it is clear that the capability of the application is to perform only coarse motion detection at the outer ring, while the second ring begins to have a reduced capability for pattern recognition, a feature fully acceptable in the first ring. These capabilities are better appreciated in Fig. 15, showing the pyramidal ROIs of the hierarchical structure, without the distortion present in the synthesized retinal structures. The capabilities are obviously better in the VGA case in Figs. 16 and 17, but the data volume is higher, too, showing the importance of selective data reduction to keep an acceptable system performance at high resolution.

The merits of an adaptive foveal imager are evident when we compare Figs. 14 and 15: If fixed structure sensors had been used to perform foveations on the person in Fig. 14, two mechanical saccades would be required to move the foveae onto the person appearing on the left of the FOV in Fig. 15. Those mechanical saccades would have changed completely the image background, which would imply a computational load for the processor not required with ARSs. Another advantage, which is also appreciated in Fig. 15, is the ability to resize the foveae (i.e., the
Figure 13. A prototype version of the stereo imager platform.

Figure 14. Top: Uniresolution images (320 × 256 pixels). Bottom: Synthesized retinal images. Foveae sizes: 26 × 46 pixels; 40° angle lenses. Left channel: l, r, t, b = 10, 11, 11, 3. Right Channel: l, r, t, b = 11, 10, 10, 4.
bounding boxes) to ensure full coverage of the region of interest, minimizing the number of foveations: If, instead of foveating onto the person in Fig. 15, with foveae higher than wide, the target had been a car appearing on the right of the FOV, the fovea should be wider than high, a simple task for ARSs, but impossible for fixed structure sensors, requiring them to perform several foveations or to try at lower resolution levels, with wider solid angle.

Also noticeable is the stereo disparity in Fig. 15, where the person approaching the imager causes foveations in different regions of each pair of images, as indicated by the differences between the sets of $l, r, t, b$ parameters. Stereo disparity is actually used only to estimate distances to the platform, based on the $l, r, t, b$ sets of the chosen ROIs and the object coordinates within them.

The advantages of multifoveal images can be appreciated in Figs. 16 and 17, which show a VGA image and a bifoveal image obtained from it. In the case of a unifoveal structure foveating onto the truck, cars at the right in Fig. 16 would fall within the outer ring of the ARS, but it would be easy to detect their motion. To analyze the car, a foveation onto it would imply stoppage of truck analysis, because it would be on the peripheral ring of the ARS foveating onto the car. However, the bifoveal structure in Fig. 17 allows both actions in parallel. It is also noticeable that processing the adaptive fovea at the right would allow recogni-
tion of the car, while ignoring the other relatively close car at the bottom-right corner.

The imager implementation in the above-mentioned FPGA and PCI-bus board is able to provide bifoveal structures from VGA images, as shown in Figs. 16 and 17. To increase the multifoveal capability, a larger FPGA is required.

VI. CONCLUSIONS

Active vision systems require complex algorithms and speed to operate properly, demanding selective data reductions to minimize the vision processor computational burden, which can also be reduced if part of it is implemented in hardware. Adaptive multiresolution, a key technique for active vision systems, allows strong data reductions, close to 90% for typical foveae, an important factor to reduce system resources, volume, weight, and power consumption, as well as to increase system portability. The negligible delay in obtaining the hierarchical data structures makes the platform suitable to process the very high frame rates provided by some new sensors, around 500 fps, although this is still a useless feature because of the comparative low speed of the vision algorithms. The wide FOV covered with multiresolution reduces the number of saccades to track a mobile, because it remains longer under the FOV. Besides that, electronic foveation makes precise and expensive mechanical pointing systems to perform saccades unnecessary.

The attention mechanisms of active vision systems are driven by scene events whose extraction may also require intensive computations. These processes may be shortened if background information is provided. For that purpose, the imager supplies an intermediate resolution background, obtained and updated after the reduced data in the second pyramid level, but enough to detect relevant events. Thus, the attention mechanisms involved in multiresolution-based applications are greatly simplified with the use of specific background extraction techniques in the platform, which, coping with the multiresolution algorithms and the improved image-processing speed, allow efficient tracking of high-speed mobiles. As with the coarse saccades needed to change the wide FOV before losing a mobile target, the reference background is lost but the imager background extractor uses the first image after the saccades to create an acceptable, updated background after a few frames, typically 8 to 12.

The compact multifoveal imager core presented in this article integrates the generation of a vision pyramid and the extraction of adaptive multiresolution data, as well as the capability to control some camera parameters—two important considerations for active vision systems. The hardware implementation of the pyramid is one of the most relevant functions of the imager. A similar construction in software would be a computationally intensive task and would therefore reduce the system speed and efficiency.

Two parallel interfaces have been used to test the imager performance under different bandwidth requirements. However, considering potential systems constraints, such as the interface cables related to PCI-bus boards, USB and 1394 (Firewire) serial interfaces.
are being designed to minimize interface wiring or to allow longer interface cable lengths—a convenient feature in some applications.

Multifoveal imagers pose specific constraints not only at the applications but also at the imager level. The number of FIFOs and their dimensions are conditioned to the structure and size of SRAM planes in FPGAs. Thus, for the multifoveal imager mentioned in subsection IV.D, with up to four foveae, a 130 KE FPGA is required, having 130 Kgates and, especially, 65 K SRAM bits to configure wider and deeper FIFOs. This requirement is not a problem but a convenient opportunity, considering the interest in increasing even more the field of view and the resolution. Progressive-scan CMOS megapixel sensors are on the market, offering the chance to increase the FOV, the fovea resolution, and the number of levels from 4 to 5, which increases the adaptive pyramidal data compression. The hardware implementation of algorithms for high-level functions of active vision systems, using megapixel sensors and larger FPGAs, is our current research activity after the results obtained with the stereo imager at some application levels.

REFERENCES


B. Zuber, Models of oculomotor behavior and control, West Palm Beach, FL: CRC Press, 1981.