A CMOS-3D Reconfigurable Architecture with
In-pixel Processing for Feature Detectors

M. Suárez, V.M. Brea, F. Pardo
Centro de Investigación en Tecnologías de la Información (CITIUS)
University of Santiago de Compostela
Santiago de Compostela, Spain
Email:manuel.suarez.cambe@usc.es

R. Carmona-Galán
A. Rodríguez-Vázquez
Instituto de Microelectrónica de Sevilla (IMSE-CNMI)
CSIC
Universidad de Sevilla
Sevilla, Spain

Abstract—This paper introduces a two-tier CMOS-3D ar-
chitecture for generation of Gaussian pyramids, detection of
extrema, and calculation of spatial derivatives in an image. Such
tasks are included in modern feature detectors, which in turn can
be used for operations like object detection, image registra-
tion or tracking. The top tier of the architecture contains the image
acquisition circuits in an array of 320 × 240 active photodiode
sensors (APS) driving a smaller array of 160 × 120 analog
processors for low-level image processing. The top tier comprises
in-pixel Correlated Double Sampling (CDS), a switched-capacitor
network for Gaussian pyramid generation, analog memories and
a comparator for in-pixel Analog to Digital Converter (ADC). The
reusae of circuits for different functions permits to have a small
area for every pixel. The bottom tier of the architecture contains a
frame buffer with a set of registers acting as a frame-buffer with a
one-to-one correspondence with the analog processors in the
first tier, the digital circuitry necessary for the extrema detection
and the calculation of the first and second spatial derivatives in
the image, as well as Harris and Hessian point detectors. For the
time being, a behavioral model of the first tier including mismatch
and feedthrough and charge injection errors is discussed. Also,
a VHDL model for the bottom tier is addressed. The two-tier
architecture is conceived for its implementation on the 130 nm
CMOS-3D technology from Tezzaron. A companion chip will
perform the higher-level operations as well as communications.
In this technology an area of 300 $\mu m^2$ per analog process has been
estimated. The architecture proposed for pyramid generation lets
a frame rate of 180 frames/s for an ADC conversion time of 120
$\mu s$. The architecture has been proved with object detection for
given feature detectors.

I. INTRODUCTION

Operations as object detection and recognition [1], image re-
trieval, image registration, or tracking rely on local properties.
Feature detectors as Harris [2], Harris-affine [3], Hessian [4],
Hessian-affine [5], Scale Invariant Feature Transform (SIFT)
[1] or Speeded Up Robust Feature (SURF) [6] identify certain
pixels in the image and make a correspondence with pixels in
other image to perform the corresponding task. Usually there exist variations as scale changes, rotations, deformations,
occlusions, etc. between pairs of images. Feature detectors
than can deal with these variations are known as invariant
feature detectors. This is the case of SIFT and SURF. The
price is computation time. Harris and Hessian feature detectors
provide higher frame rates, but less invariance or accuracy.
The approach addressed in this paper is thought as a solution
encompassing different feature detectors or modes of operation
on a CMOS-3D-based system. In the SIFT mode, the archi-
tecture runs the SIFT algorithm, giving high accuracy. In the
Harris or Hessian modes, the accuracy is reduced in exchange
of speed. The user selects the appropriate mode of operation
according to the needs of the application. The feasibility of
several feature detectors on a single chip is possible thanks to
the fact that both SIFT and Harris- and Hessian-based feature
detectors share part of the pixel-level functions.

More specifically, as a first step in SIFT, as well as Harris
and Hessian algorithms in their multiscale version, the so-
called Gaussian pyramid to achieve scale invariance is a must.
The Gaussian pyramid comprises a set of versions of the original image (octaves) resized by a factor 1/4 from one
to another. Every octave is made up of a variable amount of
scales. These scales are the result of the application of a
Gaussian filter with a certain width ($\sigma$) over the previous scale.
The number of octaves and scales depends on the application.
SIFT implements an approximation of the Laplacian operator
as Difference-of-Gaussians (DoGs), which are the difference
of two successive scales, conforming a pyramid of DoGs.
The keypoints are found on three successive DoGs in an 8-
neighborhood region around one pixel of interest (27 neigh-
bors). In the Harris- and Hessian-based methods, the keypoints
are located on the scales. The SIFT algorithm is completed with
three subsequent main steps: 1) the accurate location of
keypoints, 2) the so-called orientation assignment, and, finally
3) the generation of a descriptor vector for every keypoint
in every scale [1]. Although there is a data reduction among
the main steps, e.g. from the whole image to the number of
keypoints, claimed as usually less than 1% of the amount of
pixels in the image, still there is a high computational burden
in the higher processing stages for SIFT. This can be avoided
in simpler feature detectors which do not have the stages of
accurate location of keypoints, or the making of the descriptor
vectors, gaining speed at the expense of less accuracy or
invariance. This is the case of Harris- and Hessian-based
methods. Also, apart from the Gaussian pyramid commented
above, Harris- and Hessian-based methods share many pixel-
level operations with SIFT. For instance, in SIFT the gradient
calculation is needed for the stage of orientation assignment,
which in turn is required for the extraction of the second moment matrix (or auto-correlation matrix that provides the predominant directions of the gradient in a neighborhood of a point) used in Harris. Few additional hardware modifications allow for the second derivatives for the Hessian matrix. This work takes advantage of this fact, reusing the hardware for gradient and keypoint detection for running different feature detectors. Also, the reuse of circuits combined with CMOS-3D technology permits a pixel-per-processor approach, leading to massive parallelism at pixel-level.

This paper addresses the architecture for Gaussian pyramid generation with 320 \times 240 pixels, and a 160 \times 120 array of analog in-pixel processors or cells, as well as the hardware to realize in parallel the SIPT, Harris and Hessian operations on a CMOS-3D stack on 100 nm CMOS-3D technology from Tezzaron [7]. For the time being, this technology provides two tiers tied to a foundry-provided 1 Gb DRAM. The Gaussian pyramid generation is provided by a Switched-Capacitor (SC) network laid in the top tier [8]. This permits to use the capacitors not only for Gaussian pyramid, but also for functions like Correlated Double Sampling (CDS) during the acquisition phase, or as analog memories at pixel-level. The registers for AD conversion and for the temporary storing of data and the hardware for reading and digital processing like keypoints location is placed in the bottom tier. The registers are pitch-matched with the top tier. Several approximations should be taken for the derivatives calculation by serial digitization of the four pixels. The system proposed lets run the SIPT providing keypoints, the first derivatives (dx,dy), and Harris. The user could choose the best processing for the application. The \sigma-levels are user-selectable as well the number of scales. Taking a conversion time of 120 \mu s [10], the frame rate for processing an image with 3 octaves and six scales per octave is 180 frames/s.

II. CMOS-3D-BASED SYSTEM

The scheme of the CMOS-3D-based vision system is shown in Fig. 1. The architecture will be implemented on a 130

III. TOP TIER

A. Analog Processor Architecture

Fig. 2 shows the schematic of every cell or processor located in the top tier. The different blocks are highlighted in different colors. The acquisition is performed with a 3T Active Pixel Sensor (marked in red in Fig. 2). The state capacitors that work as analog memories are enclosed in green. The comparator of the in-pixel A/D converter is indicated in blue. Finally, part of the switched-capacitor network used for the Gaussian pyramid is labeled Switched Diff. Block A and B. The state capacitors are used for the CDS and for the Gaussian pyramid. The capacitor C of the CDS is reused for the A/D converter. This is performed with an 8-bit single-slope A/D converter. It should be noted that we opt for a 4 3T APS assignment per processor, also called cell. This is a must for a reasonable area for the massively parallel processing of our approach. At the same time, the fill-factor improves. Nevertheless, the performance decreases due to the serial acquisition and digitization of the four sensors in the first octave. A time diagram is shown in Fig. 3 for the pixel labeled s1 (\phi_{s1} and C_{s1} in Fig. 2) for the
SIFT algorithm with 3 octaves and 6 scales per octave. After the conversion, three main operations are made: Gaussian filtering, also called diffusion (signals φ_{diff}), copying of value in the comparator (φ_{write,ref}, φ_{write}, and φ_{comp,ref}), and the comparison. Note that these operations should be made in series for the four sensors.

The acquisition of the image is performed with the classical 3T-APS approach which together with the state capacitors C_{Si}, and the capacitor C make the CDS. This architecture is shown in Fig. 4(a) [9]. The result is stored in the corresponding analog memory C_{Si} after the integration period, which is given by the expression:

\[ C_{Vi} = V_{ref} + \frac{C}{C_{Si}} [V_S(t_0) - V_S(t_1)] - V_Q \]  

where \( V_{ref} \) is an analog reference signal, \( V_S(t_0) \) and \( V_S(t_1) \) are the values sensed at the sensor \( s_i \) at instants \( t_0 \) and \( t_1 \) respectively, and \( V_Q \) is the quiescent point of the inverter.

The acquisition is controled by signals \( \phi_{ref,si}, \phi_{comp,ref,si}, \phi_{acq,si} \), \( \phi_{read,si} \) and \( \phi_{write,si} \) with \( i = 1, 2, 3, 4 \). The data path is shown in Fig. 2 as a green dashed line. Disabling \( \phi_{acq,si} \) and enabling \( \phi_{write,si} \), we can read/write the value stored at the output of the inverter as:

\[ V_{Si} = V_{ref} + \frac{C}{C_{Si}} [V_Q(t_0) - V_Q(t_1)] \]  

These equations hold for a sufficiently high gain of the inverter.

The comparator for the A/D conversion is realized with the inverter and by reusing the capacitor \( C \) when the signal \( \phi_{acq} \) is turned off. The 8-bit single-slope A/D converter is distributed among two tiers: the analog ramp generator and the comparator in the top tier, and a register and a digital counter in the bottom tier. A scheme of such an A/D converter is displayed on Fig. 4(b). To carry out the conversion, firstly, the value given by Eq. (2) is written in the capacitor \( C \), enabling signals \( \phi_{write,ref}, \phi_{write} \) and \( \phi_{comp,ref} \). Secondly, this value is compared with the analog global ramp \( V_{ramp} \), by enabling \( \phi_{conv} \). After that, the output of the inverter is given by:

\[ V_{out} = -K(V_{ramp} - V_{Si}) + V_Q \]  

When the first term of Eq.(3) has a zero crossing, the comparator changes the logic value at its output. The output of the comparator is the signal that enables/disables the reading of the registers allocated in the bottom tier. This conversion signal is driven to the registers by a Through Silicon Vias (TSV). The data path of the conversion in the top tier is enclosed in a dashed blue line in Fig. 2.

The other functionality of the processors in the top tier is the Gaussian filtering, or Gaussian pyramid. This task is executed by the peripheral blocks of Fig. 2, working together with the state capacitors which again are reused. The peripheral blocks are implemented with a switched-capacitor network [8]. The switches controled by the signal \( \phi_{1/4} \) let make the downscaling 1/4 merging the value of the four state capacitors for the second octave. An in-depth explanation of the Gaussian pyramid generation is given in the next section.

B. Nominal Analysis of the Diffusion Network

The Gaussian filtering, which is needed for the Gaussian pyramid, is the solution of the heat equation. A Resistive-Capacitive (RC) Network is a natural solution of this equation. In [8] a switched-capacitor network based on a double Forward-Euler configuration was proposed. This double Forward-Euler network has the same behavior as that of a continuous-time RC network, except by the discrete exchange of charge between neighboring nodes. A scheme of a node of our switched-capacitor network is displayed on Fig.5(a). Every state capacitor is identified as \( C_{ij} \), where \( i, j \) are the coordinates in the network. The exchange of charge is made with the neighbors located along the cardinal directions. This
discrete behavior lets a simple control of the $\sigma$ level by the number of cycles of the non-overlapping signals $\phi_1$ and $\phi_2$.

The Gaussian width $\sigma$ of every cycle is fixed by the relation between the state and the exchange capacitors ($C/C_E$). In particular, the value of a node at a cycle $n$ is given by:

$$V_{ij}(n) = V_{ij}(n-1) + [V_{i-1,j}(n-1) + V_{i+1,j}(n-1) +$$
$$+ V_{ij-1}(n-1) + V_{ij+1}(n-1) - 4V_{ij}(n-1)] \frac{C_E}{1+4\varepsilon}$$

(4)

On the other hand, the value for the same node in one iteration with a discrete Gaussian kernel where only the interaction with the cardinal neighbors is considered, is modeled by Eq.(5).

$$V_{ij}(n) = V_{ij}(n-1) + [V_{i-1,j}(n-1) + V_{i+1,j}(n-1) +$$
$$+ V_{ij-1}(n-1) + V_{ij+1}(n-1) - 4V_{ij}(n-1)] \frac{\varepsilon}{\sqrt{1+4\varepsilon} 2\pi}$$

(5)

By looking at equations (4) and (5), is easy to identify the $\sigma$ level per cycle as:

$$\sigma_0 = \left(\frac{2nC}{C_E} \right)^{-1/2}$$

(6)

The application of two successive Gaussian filters or kernels with $\sigma_0$ is equivalent to a Gaussian kernel with a certain $\sigma$. This property allows our hardware, which has a level of filtering $\sigma_0$ fixed by the $C/C_E$ ratio, to approach up to a certain accuracy any Gaussian kernel by recursive filtering or application of Gaussian kernels of $\sigma_0$. The dependence of $\sigma$ with the number of cycles $\sigma = \sigma(n)$ is given by Eq.(7).

$$\sigma = \sqrt{\frac{2nC_E}{4C_E + C}}$$

(7)

The $S$ scales of every octave for the Gaussian pyramid generation are generated with the same $S$ values of $\sigma$. The network for the generation of the first two octaves is displayed on Fig. 2. Peripheral blocks A and B make the interaction with neighbors along the cardinal directions. Fig. 5 (b) and (c) show their internal structure. As we have seen before, for every state capacitor the cell has four exchange capacitors. The endings $o1$, $o2$ and $o3$ in the signals names denote the octaves where they are used. After executing the diffusions needed for the first octave, the four state capacitors $C_{s1}$ (i denotes the sensor) are merged into only one through the switches controlled by $\phi_{1/4}$, making a downsampling 1/4 of the image ($M/2 \times N/2$). In this situation, the blocks A are in short-circuit, and therefore the ratio $C/C_E$ increases by a factor 2, changing the $\sigma = \sigma(n)$ relation of the system. To preserve the value of $\sigma$, the capacitors $C_{s3}$ and $C_{s4}$ are disabled after the merging. This is performed through the switch $\phi_{1/4}$.

Fig. 6 sketches the network structure of $16 \times 16$ pixels. For the third octave this involves one more set of switches. Those are controlled by signal $\phi_{1/8}$. When $\phi_{1/8}$ is on, the values stored in four cells are averaged in only one pixel, performing the downsampling of the original image from $M/2 \times N/2$ to $M/4 \times N/4$ resolution. After the merging, only one of the four cells keeps enable (highlighted in gray color). The filtering is made between these active cells similarly to the previous octaves through the paths drawn with a dotted-line in Fig. 6.

C. Error Analysis in the Diffusion Network

Several sources of error appear in the design and manufacture processes, as the non-linearity of amplifiers by the limited operating range and finite gain, the charge injection and feedthrough errors in the switches, and the mismatch in the capacitors. The main objective of this section is to make an analysis of these effects on the switched-capacitor network proposed here.

The main effect of mismatch in the switched-capacitor network is the spread of the capacitor values with respect to the nominal values $C$ and $C_E$. The error in the interaction of one pixel with one neighbor for a cycle $n$ is given by Eq.(8).

$$\xi_1(n) = \frac{C_E}{C_E + C} [\xi_1(n-1) - \xi_2(n-1)] + [V_1(n-1) - \xi_1(n-1) - V_2(n-1) - \xi_2(n-1)] \frac{1}{C_E + C} [\frac{C_E C_E - C_E C}{C_E + C}]$$

(8)

$V_1(n-1)$ and $V_2(n-1)$, and $\xi_1(n-1)$ and $\xi_2(n-1)$ are, respectively, the nominal voltages and the deviation values with respect to the nominal value in the previous scale. For $n = 1$, $\xi_1(0) = \xi_2(0) = 0$. The random deviation of values along the array means that the Gaussian kernel is pixel-depdendent. As a result, a non isotropic diffusion might come out, having some pixels with more smoothing than others. Also, the $\sigma$ dependence with the number of cycles might change. Fig.7 shows the effect of the spread in the values of the state capacitors on the relation of $\sigma$ with the number of
clock cycles \( (n) \). The relation \( \sigma = \sigma(n) \) is found by comparing the result of applying \( n \) times a Gaussian kernel with spread with a conventional Gaussian kernel using a behavioral model in MATLAB. The \( \sigma \) that minimizes the RMSE between the two images is the \( \sigma \) implemented by our switched-capacitor network. Simulations of 50 random normal distributions with a standard deviation of \( \sigma = \sqrt{C} \) were run. As seen, the effect of mismatch on the relationship \( \sigma = \sigma(n) \) is really small over the whole image.

Other important error sources are the feedthrough and the charge injection from the switching. The first is due to the coupling of the control signal through the overlapping capacitance between a switch and the capacitor driven by such a switch, e.g. the switches connecting the state and the exchange capacitor. The second effect is due to the injection of the charge accumulated in the channel during inversion that goes out through the source and drain terminals when the switch turns off. As Fig.4(a) shows, in the switched-capacitor network when one switch turns off, the switch turns on at the same node. In this way, the effects of one of the switches are cancelled out by the other one in a first order consideration. Second order effects like mismatch between switches and dynamic effects appear, rendering a non-zero contribution.

D. Area Estimate

The circuit sketched in Fig. 2 contains 4 APS, 2 inverters of gain \(-K\), 4 state capacitors, an additional capacitor for offset-cancellation in the comparator used for ADC, 16 exchange capacitors, and around 60 switches. We can give an area estimate per pixel if we account for: 1) an area of \( 5 \mu m \times 5 \mu m \) per photodiode, 2) capacitors of 100 \( fF \), which have a density of \( 1 fF/\mu m^2 \) for the 150 nm CMOS-3D Tezzaron technology, 3) exchange capacitors of 10 \( fF \), 4) double-cascode inverters to enhance a high enough gain (\( K > 60 \, dB \)) to reduce errors in closed-loop configurations, amounting to \( 50 \mu m^2 \) if we take the implementation presented in [7] as a reference (PDSOI 150 nm), 5) 2 \( \mu m^2 \) per switch, and 6) a TSV of \( 5 \mu m^2 \). All in all it yields around \( 250 \mu m^2 \) per pixel, which we overestimate up to \( 300 \mu m^2 \), accounting for the routing. These numbers would lead to \( 23 \, mm^2 \) for an image of QVGA resolution. It should be noted that the ratio \( C/C_k \) chosen was \( 100 fF/10 fF \). This ratio was employed for the graph of Fig. 5, which permits a \( \sigma \) range from 0.45 to 3, enough for SIFT-based applications.

E. SIFT-based Assessment: Object Detection

To test the validity of the switched-capacitor network and its robustness to mismatch errors, a behavioral model was implemented in MATLAB. In a first order we assume that both charge injection and feedthrough errors are canceled due to the switching mechanism of the network, as it was discussed in Section III-C. The behavior model was tested through object detection. For that we employed the SIFT implementation available at [11], assessing the so-called precision and recall for the cases of images as shown in Fig. 8(a), in which we can see an object and its rotated version of \( 45^\circ \). The precision is defined as \( p = tp/(tp+fp) \), and recall is \( r = tp/(tp+fn) \), with \( tp \) the number of true positives, \( fp \) the number of false positives and \( fn \) the number of false negatives. \( tp + fp \) is the number of matches, shown as overlapping points in Fig. 8(b). The matches are calculated by comparing the descriptor vectors of two keypoints through Euclidean distance. If this distance is below (above) a certain threshold \( (th) \), the corresponding pair of keypoints is a match. A match becomes \( tp \) when it also complies with the location condition; otherwise it is a false positive. The location condition can be checked easily in a known transformation as that of Fig. 8. It is also possible to calculate \( fn \). Fig. 8(b) was obtained for several thresholds \( th \). In this case, random normal distributions with standard deviation \( \sigma = \sqrt{C} \) were run. As it can be seen, the shape of the curve with the switched-capacitor networks subject to mismatch resembles that of the nominal one, but with little worse performance. The application dictates whether or not the mismatch is detrimental.

IV. BOTTOM TIER

A. Digital Domain Processing

As it was said before, the A/D converter is shared among the two tiers. We opt for an 8-bit single-slope A/D converter, as it is the best option in terms of area per processor. If we distribute the comparators in the top tier, and the signals of a global counter (digital code generator) along the registers in the bottom tier, just one TSV is needed for a set of 4 pixels, namely for what we call a cell. The drawback of the single-slope A/D converter is the long time processing, which, based on data extracted from [10], we have estimated in 120 \( \mu s \) per
conversion. The same array structure is repeated in the bottom tier, making it easier to have pitch-matched cells between the top and bottom tiers. Thus, the digitized pixels are written to an $M/2 \times N/2$ set of registers. Each one of these sets of registers contains 6 8-bit registers. Two of them make the conversion of the scale $k$ in conjunction with the comparator of the top tier (with $k$ indicating the scale in a given octave). Two registers are needed to let the conversion of one pixel, while the others are reading for further processing. The remaining registers store the four values of the previous scale ($k-1$). This way the whole $M \times N$ image is stored in the bottom tier. We name these 4 pixels as $P_1, P_2, P_3$ and $P_4$, which correspond with locations $(i,j)$, $(i, j+1)$, $(i+1,j)$ and $(i+1,j+1)$, respectively, where $i$ is for rows and $j$ for columns. The four pixels $P_1$-$P_4$ are digitized in series, as there is only one Through-Silicon-Vias (TSV) per every 4-pixel processor. This means that all pixels $P_1$ are digitized in one conversion cycle, $P_2$ in a second conversion cycle, and so on for pixels $P_3$ and $P_4$. Therefore, four serial conversion cycles are needed for the digitization of the whole image in the first octave.

Fig. 9 outlines the sequence of operations run in the CMOS-3D stack. Fig. 10 depicts the architecture of the circuit located in the bottom tier of the CMOS-3D stack. The frame buffer is an $M/2 \times N/2$ set of registers to store the different scales of the Gaussian pyramid. After every diffusion or Gaussian filtering of the image and its digitization, the bottom tier works. Several operations are run in parallel: 1) the digitization of pixel $P_1$ at scale $k$, 2) scales $S(k)$, 3) difference of Gaussian between scales $k$ and $k-1$, namely $DoG(k)$, 4) horizontal gradient along the $x$ and $y$ direction for scale $k$, $dx(k)$ and $dy(k)$, 5) Harris and Hessian keypoint detection over scales and 6) Harris and detection Hessian over DoGs. Subsequently, the results are sorted out in groups of 128 bits (16 words of 8 bits each) and transfer in burst mode to the DRAM memory.

The images from the buffer array are read in groups of 20 registers (16 actual pixels + 4 for windowing purposes) row by row in order to provide the 16 first and 16 second derivatives can be made at the same time. For every row $i$, the 20 columns of pixels $P_i$ are selected through the multiplexers seen in Fig. 10. Four multiplexers are needed for this task. Two of them are shared by the first and the second octaves for scales $k$ and $k-1$. Both scales are required in the $DoG$ calculation. The two other multiplexers are employed for the third octave. It should be noted that for the first and the second octave, the multiplexers can be shared, as we always access all the registers along a row. In the case of the first octave we need 4 “cycle” readings to transfer pixels $P_1$-$P_4$. In the second octave and beyond, we do such a transfer in only one cycle due to the 1/4 downscaling. The lowest frequency needed for reading and doing all these operations is set by the first octave, being 10 Mhz in order to read all the pixels $P_i$ in less than 120 $\mu$s, which is the time for the A/D conversion (Fig. 9).

The gradient calculation is a very common operation in image processing. Moreover, the first derivatives are used in subsequent tasks as orientation and vector descriptor of every keypoint in the SIFT algorithm, for this reason we include this functionality in our architecture. Also, the first derivatives can be used for the Harris detector. This fact permits to share hardware resources in Harris and SIFT.

In our architecture because of the assignment of 4 pixels to one processor in the top tier the reading mechanism does not permit to yield the first derivatives $dx$ and $dy$ of one pixel at one cycle. This drawback is circumvented by calculating the gradient along a different set of axes which have been rotated 45° with respect to the conventional $x$ and $y$ axes. The gradient is now calculated by the next set of equations:

$$d_x(i,j) = I(i+1,j+1) - I(i-1,j-1) \quad (9)$$
$$d_y(i,j) = I(i+1,j-1) - I(i-1,j+1) \quad (10)$$
Fig. 11 illustrates this procedure. Fig. 11(b) shows the spatial arrangement of digitized values for the octave one. In the example, only the pixels $P_1$ (grey) are accessible. Fig. 11(c) shows the block diagram for gradient, Harris and Hessian extraction. As shown in Fig. 11(b), groups of 20 pixels are transferred row by row from the frame buffer to this block. This process is showed with an 'X' (gradient in blue and second derivatives in red) in Fig. 11(b). The derivatives cannot be obtained at the border of the image due to the neighborhood dependence. These points are represented as "O" in the same figure. The results of the Harris and Hessian are two images of $M \times N$ sizes of 1 and 2 bits per pixel, respectively. "1" means an extrema and "0" is a point without significant information for SIFT. The Harris algorithm has three states: "00" is a corner, "01" an edge and "1X" a flat. The block of Fig. 11 makes one calculation by cycle (one pixel). The frequency of this block should be 160 MHz ($16 \times 10$ MHz) 16 times higher than the reading frequency to the processing of the 16 words of a string.

Other detectors are based on the localization of characteristic points through the Hessian matrix. This matrix needs the second derivatives. The calculation of such derivatives requires the neighbors around a point in a 4-neighborhood. As it was mentioned before, in our reading mechanism, a given pixel does not have the right neighbors to perform the first derivatives along the conventional x and y axes (horizontal and vertical directions). It would be possible, however, to do such an operation with pixels located two pixels apart. An approach to the second derivative is made by generating the neighbors located one pixel apart along the horizontal and vertical directions by interpolating the pixels located two pixels apart from the one under study. Thus, in this procedure, the neighbor at $(i+1, j)$ is generated as $I(i+1, j) = \frac{I(i+2, j) + I(i, j)}{2}$. With this approximation the second derivatives are given by Eq. (12-13).

$$d_{xx}(i, j) = I(i, j + r)) + I(i, j - u) - v \cdot I(i, j)$$  \hspace{1cm} (11)
$$d_{yy}(i, j) = I(i + u, j) + I(i, j - u) - v \cdot I(i, j)$$  \hspace{1cm} (12)
$$d_{xy}(i, j) = I(i + u, j + v) + I(i + u, j - v) + I(i - u, j + v) + I(i - u, j - v) - 2 \cdot v \cdot I(i, j)$$  \hspace{1cm} (13)

where $u = 2$ and $v = 1$ for the first octave, and $u = 1$ and $v = 2$ for the next octaves, given that at the second and the third octaves every pixel has the right neighbors along horizontal and vertical directions to perform the gradient along the conventional x and y axes, and thus the approach with the interpolation is not needed.

Fig. 12 displays the circuits of every set of registers in the buffer array. Every set of registers comprises 6 8-bit registers to store pixels $P_1$-$P_4$. As said before, there is only one TSV connecting the two tiers. This is a 1-bit signal driving two AND gates with $\phi_{con13}$ and $\phi_{con14}$ as inputs, yielding the enable signals for the top two registers, $R_{13,K}$ and $R_{24,K}$. The top two registers store the pixels of scale $k$. The four bottom registers keep pixels $P_1$-$P_4$ for scale $k-1$. Scales $k$ and $k-1$ are available on the corresponding buses of every set of registers for DoG calculation. The sequence of operations to achieve every scale in the first octave is as follows: Pixel $P_1$
is digitized into register $R_{13,K}$ with $\phi_{\text{conv}13}$ on. Subsequently, pixel $P2$ is digitized and stored in register $R_{24,K}$, following a similar process with signal $\phi_{\text{conv}24}$ on. During $\phi_{\text{conv}24}$ on the DoG for all pixels $P1$ of scale are calculated and written into the DRAM. After the reading of pixels $P1$ the content of register $R_{13,K}$ is transferred into register $R_{1,K}$ by means of signal $\phi_{W1}$ on. Later on, pixels $P3$ are digitized in register $R_{13,K}$ while the pixels $P2$ are read, and the process continues up to pixels $P4$, completing the first octave.

B. Synthesized Data

The frequency specifications in our system are set by the operations run in parallel during the A/D conversion of a given pixel (see Fig. 9). The A/D conversion time per pixel was estimated at 120 $\mu$s. During this time, pixels $P_i - 1$, which have already been A/D-converted, are being written into the bottom DRAM. Also, the DoGs, $dx$ and $dy$ calculations and their storage in the bottom DRAM for pixels $P_i - 1$ are being performed at the same time as the A/D-conversion of $P_i$. Thus, two types of operations are run in parallel, namely, data calculation and memory-writing. The time it takes to perform both in series should be inferior to 120 $\mu$s.

In our design, the DoGs, $dx$ and $dy$ are calculated and stored in two groups of 16 pixels, so for each pixel $P_i$, the total number of clock cycles is given by: $(M \times N)/16 \times 4$. In a VGA image, this renders a minimum clock frequency of 10 MHz. These numbers are not hard to achieve on a modern CMOS technology. As an example, our circuit has been synthesized on a Virtex-6 from Xilinx, reaching 375 MHz. This frequency would lead to less than 12 $\mu$s for the DoGs, $dx$ and $dy$ in a VGA image. Still, 108 $\mu$s would be required for the memory storage of DoGs, $dx$ and $dy$.

The memory writing is set by the specifications of the DRAM provided by Tezzaron. In this case, the memory contains 8 I/O ports with 128 bits each. Every port supports 256 bits per cycle at 1GHz, which yields a data transfer rate $TR = 256 \times 10^9$ bytes/s at every port. For a given type of pixels $P_i$ in a image $320 \times 240$ image, our system demands transfer rate of 56 Gbits/s. As seen, the data transfer rate of the DRAM provided by Tezzaron meets the needs of our application.

Concerning area, it should be said here that a synthesis on an FPGA has been made. Nevertheless, the FPGA resources do not match naturally the implementation presented here, leading to a misleading large area occupation. Custom solutions like the one reported in [10] yield an area of 50 $\mu$m $\times$ 50 $\mu$m for a set of 6 8-bit registers in 150 nm FDSOI technology, which can be taken as a baseline for the frame buffer.

V. CONCLUSIONS

This paper has addressed the architecture of a CMOS-3D-based vision system for running different feature detectors. The system is thought as an approach where the user can select the most appropriate feature detector according to the needs of the application. The architecture executes two main modes: 1) SIFT mode, providing high accuracy at the cost of low speed, and 2) Harris and Hessian feature detectors, yielding speed in exchange of worse accuracy. Both modes are possible due to: 1) the CMOS-3D architecture, and 2) the fact that running SIFT implies to run some of the operations required for Harris- or Hessian-based algorithms. The work also introduces a new pixel architecture with in-pixel CDS, and in-pixel A/D conversion by means of an 8-bit single-slope A/D converter. The reuse of different circuits permits to have a reasonable area for every pixel. Also, the architecture presents an assignment of 4 3T APS per processor, rendering massively parallel processing, very adequate for operations at pixel-level, quite abundant in any feature detector. The architecture is implemented with a two tier CMOS-3D stack. The top tier contains the pixels. Every pixel is completed with the circuits needed for a switched-capacitor network. Such a network gives Gaussian filtering, needed for many feature detectors. The paper presented here has addressed a behavioral model of the top tier with manufacture errors included. The feasibility of the implementation is proven with object detection by SIFT. The bottom tier contains a frame buffer and digital circuits for further processing. The frame buffer is not only used for image storage, but it is also used for A/D conversion. An area of 300 $\mu$m$^2$ per pixel was estimated on the 130 nm CMOS-3D technology from Tezzaron. The CMOS-3D stack is tied to a 1Gb DRAM provided by the foundry. The circuit addressed in this paper permits to extract the pyramid, derivatives, Harris keypoints, and the DoG keypoints with a frame rate of 180 frames/s.

VI. ACKNOWLEDGMENT

This work has been funded by Xunta de Galicia (Spain) and MICYT (Spain) through projects 10PXIB206037PR and TEC2009-12686.

REFERENCES