Focal-plane dynamic texture segmentation by programmable binning and scale extraction

J. Fernández-Berni and R. Carmona-Galán

Abstract Dynamic textures are spatially-repetitive time-varying visual patterns that present, however, some temporal stationarity within their constituting elements. In addition, their spatial and temporal extents are *a priori* unknown. This kind of patterns is very common in nature, therefore dynamic texture segmentation is an important task for surveillance and monitoring. Conventional methods employ optic flow computation though it represents a heavy computational load. Here we describe texture segmentation based on focal-plane space-scale generation. The programmable size of the subimages to be analyzed and the scales to be extracted encode sufficient information from the texture signature to warn its presence. A prototype smart imager has been designed and fabricated in $0.35\mu\text{m}$ CMOS, featuring a very low-power scale-space representation of used-defined subimages.

1 Introduction

Dynamic textures (DTs) are visual patterns with spatial repeatability and a certain temporal stationarity. They are time-varying, but some relations between their constituting elements are maintained through time. Because of this, we can talk about the frequency signature of the texture [1]. An additional feature of a DT is its indeterminate spatial and temporal extent. Smoke, waves, a flock of birds or tree leaves swaying in the wind are some examples. The detection, identification and tracking of DTs is essential in surveillance because they are very common in natural scenes. Amongst the different methods proposed for dynamic texture recognition, those based in the optical flow are currently the most popular [2]. Optic flow is a

J. Fernández-Berni and R. Carmona-Galán
Institute of Microelectronics of Seville (IMSE-CNM-CSIC)
Consejo Superior de Investigaciones Científicas-Universidad de Sevilla
C/ Americo Vespucio s/n 41092 Seville (Spain)
e-mail: {jfberni,rcarmona}@imse-cnm.csic.es

computationally efficient and natural way to characterise the local dynamics of a temporal texture. This is the case for weak DTs, which become static when referred to a local coordinate system that moves across the scene. However, the recognition of strong DTs implies a much greater computational effort. For these textures, possessing intrinsic dynamics, the brightness constancy assumption associated to standard optical flow algorithms cannot be applied. More complex approaches must be considered in order to overcome this problem. Recently, interesting results have been achieved by applying the so-called brightness conservation assumption [3]. However, this method means heavy computational load and the subsequent high energy consumption. For a particular type of artificial vision systems, a power-efficient implementation of dynamic texture recognition is mandatory. Wireless multimedia sensor networks [4] is an obvious example. These networks are composed of a large number of low-power sensors that are densely deployed throughout a region of interest in order to capture and analyse video, audio and environmental data from their surroundings. The massive and scattered deployment of these sensors makes them quite difficult to service and maintain. Therefore, energy efficiency must be a major design goal, in order to extend the lifetime of the batteries as much as possible.

We propose an approach that do not rely in heavy computation by a general-purpose processor, but on an adapted architecture in which the more tedious tasks are conveyed to the focal plane to be realized concurrently with the image capture. This results in a simplified scene representation that carries, nevertheless, all the necessary information. In this scheme, redundant spatial information is removed at the earlier stages of the processing by means of simple, flexible and power-efficient computation at the focal plane. This architecture encode the major features of the DTs, in the sense that the spatial sampling rate and the spatial filter passband limits are programmed into the system. This permits to track textures of an expected spatial spread and frequency signature. The main processor operates then on a reduced representation of the original image obtained at the focal plane, thus its computational load is greatly alleviated.

2 Simplified scene representation

2.1 Programmable binning and filtering

In general, existing research on dynamic textures recognition is based on global features computed over the whole scene. A clear sign of this fact is that practically all of the sequences composing the reference database *DynTex* [5] contain only close-up sequences. It does make sense, in these conditions, to apply strategies of global feature detection. However, in a different context, e. g. video-surveillance, textures can appear at any location of the scene. Local analysis is required for texture detection and tracking. One way of reducing the amount of data to be processed is to summarize the joint contribution of a group of pixels to the appropriate medium-

Fig. 1 Binning and filtering applied to a scene containing a flock of starlings

size-feature index. Let us consider the picture in Fig. 1(a), which depicts a flock of starlings. It is known that these flocks maintain an internal coherence based on local rules that place each individual bird at a distance prescribed by their wing span [6]. This is an example of self-organized collective behaviour, whose emergent features are characterized by a set of physical parameters like, for instance, flock density. We can estimate the density of the flock—more precisely, the density of its projected image—by conveniently encoding the nature of the object into the spatial sampling rate and the passband of the spatial filter selected to process the subimages.

The first step is then to subdivide the image into pixel groups that, for the sake of clarity, will be of the same size and regularly distributed along the two image dimensions (Fig. 1(b)). Then, if the image size is $M \times N$ pixels, and the image is divided into equally sized subimages, or bins, of size $m \times n$ pixels, we will end in a representation of the scene that is $1/R$ times smaller than the original image, being:

$$R = \left(\frac{m}{M}\right) \left(\frac{n}{N}\right) \quad (1)$$

The problem is conveyed now to finding a magnitude that summarizes the information contained in each $m \times n$ -pixel bin and is still useful for our objective of texture detection. In the case of the starling flock density, a measure of the number of birds contained in a certain region of the image can be given by the high-frequency content of each bin. Notice that features of a low spatial frequency do not represent any object of interest, i. e. bird, but details belonging to the background. Therefore the value of each bin, represented in Fig. 1(c), can be defined as the quotient:

$$B_{kl} = \frac{\sum_{\forall |\mathbf{k}| > 0} E_{kl}(\mathbf{k})}{\sum_{\forall \mathbf{k}} E_{kl}(\mathbf{k})} \quad (2)$$

where $k \in \{1, \dots, M/n\}$ and $l \in \{1, \dots, N/n\}$. Also, $\mathbf{k} = (u, v)$ represents the possible wave numbers and each summand $E_{kl}(\mathbf{k})$ is the energy associated with frequency $\mathbf{k}$, computed within the bin indexed by (k, l) . This is an spatial highpass filter normalized to the total energy associated with the image bin. If V_{ij} is the value of the pixel located at the i -th row and j -th column of the bin, and also, $\hat{V}_{uv}$ is the component of the spatial DFT of the bin corresponding to u and v reciprocal lengths, the total image energy associated with the bin is:

$$\sum_{\forall \mathbf{k}} E_{kl}(\mathbf{k}) = \sum_{u=0}^{m-1} \sum_{v=0}^{n-1} |\hat{V}_{uv}|^2 = \sum_{i=1}^m \sum_{j=1}^n |V_{ij}|^2 \quad (3)$$

The result is an estimation of the bird density, at a coarser grain than the full-size image, that avoids pixel-level analysis.

2.2 Linear diffusion as a scale-space generator

As already described, apart from the appropriate binning, that sets the interesting feature size and geometrical ratio, a suitable filter family is needed to discriminate the spatial frequency components within each subimage. Let us consider that an image is a continuous function evaluated over the real plane, $V(x, y)$, that assigns a brightness value to each point in the plane. If this brightness is regarded as the concentration of a scalar property [7], the flux density is proportional to the gradient and directed against it. The diffusion equation:

$$\frac{\partial V(x, y, t)}{\partial t} = D \nabla^2 V(x, y, t) \quad (4)$$

follows from continuity considerations, i. e. no sources or sinks of brightness are present in the image plane. The original image is assumed to be the initial value at every point, $V(x, y, 0)$. Then, applying the Fourier transform to this equation and solving in time:

$$\hat{V}(\mathbf{k}, t) = \hat{V}(\mathbf{k}, 0) e^{-4\pi^2 |\mathbf{k}|^2 D t} \quad (5)$$

what represents, in the reciprocal space, the transfer function of a Gaussian filter:

$$G(\mathbf{k}; \sigma) = e^{-2\pi^2 \sigma^2 |\mathbf{k}|^2} \quad (6)$$

whose smoothing factor is related to the diffusion duration t through:

$$\sigma = \sqrt{2Dt} \quad (7)$$

Fig. 2 Gaussian filters of increasing σ

Fig. 3 Resistor network supporting linear diffusion (a) and its MOS-based counterpart (b)

Hence, the larger the diffusion time, t , the larger the smoothing factor, σ , thus, the narrower the transfer function (Fig. 2), and so, the smoother the output image. Gaussian filters are equivalent to the convolution with Gaussian kernels, $g(x, y; \sigma)$, of the reciprocal widths:

$$g(x, y; \sigma) * V(x, y) = \frac{1}{2\pi\sigma^2} \int_{-\infty}^{\infty} \int_{-\infty}^{\infty} V(x-x', y-y') e^{-\frac{x'^2+y'^2}{2\sigma^2}} dx' dy' \quad (8)$$

These kernels hold the scale-space axioms: linearity, shift invariance, semi-group structure, and not enhancement of local extrema. This makes them unique for scale-space generation [8]. Scale-space is a framework for image processing [9] that makes use of the representation of the images at multiple scales. It is useful in the analysis of the image, e. g. to detect scale-invariant features that characterize the scene [10]. Different textures are noticeable at a particular scale, ξ , which is related with the smoothing factor:

$$\xi = \sigma^2 \quad (9)$$

However, convolving two large images or, alternatively, computing the FFT, and the inverse FFT, of a given image with a conventional processing architecture represents a very heavy computational load. We are interested in a low-power focal-plane operator able to deliver an image representation at the appropriate scale.

2.3 Spatial filtering by a resistor network

Consider a grid composed of resistors like that depicted in in Fig. 3(a). Let $V_{ij}(0)$ be the voltages stored at the grounded capacitors attached to each node of the resistor grid. If the network is allowed to evolve, at every time instant each node will satisfy:

$$\tau \frac{dV_{ij}}{dt} = -4V_{ij} + V_{i+1,j} + V_{i-1,j} + V_{i,j+1} + V_{i,j-1} \quad (10)$$

where $\tau = RC$. Applying the spatial DFT, for a grid of $M \times N$ nodes, we arrive to:

$$\tau \frac{d\hat{V}_{uv}}{dt} = -4 \left[\sin^2 \left(\frac{\pi u}{M} \right) + \sin^2 \left(\frac{\pi v}{N} \right) \right] \hat{V}_{uv} \quad (11)$$

Notice that, $\hat{V}_{uv}$ represents the discrete Fourier transform of V_{ij} , that is also discrete in space. Therefore, u and v take discrete values ranging from 0 to $M - 1$ and $N - 1$ respectively. Solving Eq. (11) in the time domain, we obtain:

$$\hat{V}_{uv}(t) = \hat{V}_{uv}(0) e^{-\frac{4t}{\tau} [\sin^2(\frac{\pi u}{M}) + \sin^2(\frac{\pi v}{N})]} \quad (12)$$

what defines a discrete-space version of the Gaussian filter in Eq. (6) given by:

$$G_{uv}(\sigma) = e^{-2\sigma^2 [\sin^2(\frac{\pi u}{M}) + \sin^2(\frac{\pi v}{N})]} \quad (13)$$

where now $\sigma = \sqrt{2t/\tau}$. This function approximates quite well the continuous-space Gaussian filter at the lower frequencies and close to the principal axes of the reciprocal space. At higher frequencies, specially at the bisectrices of the axes, i. e. when u and v both become comparable to M and N respectively, isotropy degrades as the approximation of $\sin^2(\pi u/M)$ and $\sin^2(\pi v/N)$ by $(\pi u/M)^2$ and $(\pi v/N)^2$, respectively, becomes too coarse.

3 VLSI implementation of time-controlled diffusion

3.1 MOS-resistor grid design

Resistive networks are, as we have already seen, massively parallel processing systems that can be employed to realize spatial filtering [11]. But a true linear resistive grid is difficult to implement in VLSI. The low sheet resistance exhibited by the most resistive materials available in standard CMOS renders too large areas for the necessary resistances. A feasible alternative is to employ MOS transistors to replace the resistors one by one. They can achieve larger resistances with less area than resistors made of polysilicon or diffusion strips. In addition, controlling the gate voltage, their resistance can be modified. They can also be operated as switches, thus configuring the connectivity of the network. This substitution of resistors by MOS transistors, however, entails, amongst others, linearity problems. In [12], the linearity of the currents through resistive grids is achieved by means of using transistors in weak inversion. The value of the resistance associated to each transistor is directly controlled by the corresponding gate voltage. This property of current linearity is also applicable even if the transistors leave weak inversion as long as all of them share the same gate voltage [13]. Linearity is not so easy to achieve when signals are encoded by voltages as in Fig. 3(a). The use of MOSFETs operating in the ohmic region instead of resistors is apparently the most simple option [14].

Fig. 4 2-node ideal resistive grid (a) and its MOS-based implementation (b)

However, the intrinsic nonlinearity in the I-V characteristic leads to more elaborated alternatives for the cancellation of the nonlinear term [15], even to transconductor-based implementations [16].

For moderate accuracy requirements, though, the error committed by using MOS transistors in the ohmic region can be kept under a reasonable limit if the elementary resistor is adequately designed. For an estimation of the upper bound of this error, let us compare the circuits in Fig. 4. They represent a 2-node ideal resistor grid and its corresponding MOS-based implementation. The gate voltage V_G is fixed and we will assume, without loss of generality, that the initial conditions of the capacitors fulfill $V_1(0) > V_2(0)$, being $V_1(0) = V_1'(0)$ and $V_2(0) = V_2'(0)$. We will also assume that the transistor is biased in the triode region for any voltage at the drain and source terminals, that will range from V_{min} to V_{max} . The evolution of the circuit in Fig. 4(a) is described by this set of ODEs:

$$\begin{cases} C \frac{dV_1}{dt} = -\frac{V_1(t) - V_2(t)}{R} \\ C \frac{dV_2}{dt} = \frac{V_1(t) - V_2(t)}{R} \end{cases} \quad (14)$$

while the behaviour of the circuit in Fig. 4(b) is described by:

$$\begin{cases} C \frac{dV_1'}{dt} = -G_M [V_1'(t) - V_2'(t)] \\ C \frac{dV_2'}{dt} = G_M [V_1'(t) - V_2'(t)] \end{cases} \quad (15)$$

by making use of the following transconductance:

$$G_M = k_n S_n \{ 2(V_G - V_{T_n}) - [V_1'(t) + V_2'(t)] \} \quad (16)$$

where $k_n = \mu_n C_{ox}'/2$ and $S_n = W/L$. This transconductance remains constant during the network evolution, if we neglect the substrate and other second order effects, as the sum $V_1'(t) + V_2'(t)$ remains the same because of charge conservation. Therefore, and given that the charge extracted from one capacitor is ending up in the other, we can define the error in the corresponding node voltages as:

$$\begin{cases} V_1'(t) = V_1(t) + \varepsilon(t) \\ V_2'(t) = V_2(t) - \varepsilon(t) \end{cases} \quad (17)$$

or, equivalently:

$$\varepsilon(t) = \frac{V_1'(t) - V_2'(t)}{2} - \frac{V_1(t) - V_2(t)}{2} \quad (18)$$

Because of our initial assumptions, $V_1(0) = V_1'(0)$ and $V_2(0) = V_2'(0)$, we have that $\varepsilon(0) = 0$. Also, the stationary state, reached when $t \rightarrow \infty$, renders $\varepsilon(\infty) = 0$, as $V_1(\infty) = V_2(\infty)$ and $V_1'(\infty) = V_2'(\infty)$. Therefore, there must be at least one point in time, let us call it t_{ext} in which the error reaches an extreme value, either positive or negative. In any case the time derivative of the error:

$$\frac{d\varepsilon}{dt} = \frac{1}{\tau} [V_1'(t) - V_2'(t)] (1 - G_MR) - \frac{2\varepsilon(t)}{\tau} \quad (19)$$

must cancel in t_{ext} , resulting in an extreme error of:

$$\varepsilon(t_{ext}) = \frac{1}{2} [V_1'(t_{ext}) - V_2'(t_{ext})] (1 - G_MR) \quad (20)$$

Notice that for $G_MR = 1$ the error is zero at any moment. This happens if the transistor aspect, S_n , is selected to match the resistance R through:

$$S_n = \frac{1}{k_n R \{2(V_G - V_{Tn}) - [V_1'(0) + V_2'(0)]\}} \quad (21)$$

and the current values of $V_1'(0)$ and $V_2'(0)$ add up to the same $V_1'(0) + V_2'(0)$ with which S_n was selected. Unfortunately, this will very seldom happen. And because we do not know a priori where within the interval $[V_{min}, V_{max}]$ are $V_1'(0)$ and $V_2'(0)$, neither $V_1'(t_{ext})$ and $V_2'(t_{ext})$, we are interested in a good estimate of the largest possible error within the triangle $\triangle ABC$ (Fig. 5), delimited by points $A : (V_{min}, V_{min})$, $B : (V_{max}, V_{min})$, $C : (V_{max}, V_{max})$. This is because one of our initial assumptions was that $V_1'(0) > V_2'(0)$, and this condition is maintained until these voltage identify at the steady state.

Let us express this error, $\varepsilon_x = \varepsilon(t_{ext})$, as a function of $V_{1x}' = V_1'(t_{ext})$ and $V_{2x}' = V_2'(t_{ext})$. It can be done because the sum $V_1'(t) + V_2'(t)$, present in the definition of G_M , is constant along the evolution of the network, therefore also at t_{ext} :

$$\varepsilon_x(V_{1x}', V_{2x}') = \frac{1}{2} (V_{1x}' - V_{2x}') \{1 - k_n S_n R [2(V_G - V_{Tn}) - (V_{1x}' + V_{2x}')] \} \quad (22)$$

Then, any possible extreme of $\varepsilon_x(V_{1x}', V_{2x}')$ must be at a critical point, i. e. a point in which $\nabla \varepsilon_x(V_{1x}', V_{2x}') = 0$. But the only one that can be found in $\triangle ABC$ is a saddle point. Therefore, we can only talk of absolute maxima or minima, and they will be at the borders of the triangle (Fig. 5). More precisely at sides AB and BC , given that $\varepsilon_x \equiv 0$ along side AC , at points:

$$\begin{cases} V'_{1x} = V_{max} & V'_{2x} = V_G - V_{T_n} - \frac{1}{2k_n S_n R} \\ V'_{1x} = V_G - V_{T_n} - \frac{1}{2k_n S_n R} & V'_{2x} = V_{min} \end{cases} \quad (23)$$

and their values are:

$$\begin{cases} \varepsilon_x|_{max} = \frac{1}{2}k_n S_n R \left(V_{max} - V_G + V_{T_n} + \frac{1}{2k_n S_n R} \right)^2 \\ \varepsilon_x|_{min} = -\frac{1}{2}k_n S_n R \left(V_{min} - V_G + V_{T_n} + \frac{1}{2k_n S_n R} \right)^2 \end{cases} \quad (24)$$

Notice that increasing or decreasing S_n have antagonistic effects in the magnitude of $\varepsilon_x|_{max}$ and $\varepsilon_x|_{min}$ (Fig. 5). Therefore the optimal design is obtained for:

$$S_n = \frac{1}{k_n R [2(V_G - V_{T_n}) - (V_{max} + V_{min})]} \quad (25)$$

what minimizes the maximum error, rendering [17]:

$$\min(\max|\varepsilon_x|) = \frac{1}{16} \frac{(V_{max} - V_{min})^2}{V_G - V_{T_n} - \frac{V_{max} + V_{min}}{2}} \quad (26)$$

Fig. 5 Error estimate for the possible and reasonable values of S_n

It is important to notice that the design space is limited by the extreme values of S_n , those beyond which the target resistance R fall outside the interval of resistance values that can be implemented within the triangle $\triangle ABC$, that represents all the possible values that can be taken by V'_{1_x} and V'_{2_x} . These extrema are:

$$\begin{cases} S_{n_{min}} = \frac{1}{2k_n R (V_G - V_{Tn} - V_{min})} \\ S_{n_{max}} = \frac{1}{2k_n R (V_G - V_{Tn} - V_{max})} \end{cases} \quad (27)$$

Notice that if one choose to select $S_n = (S_{n_{min}} + S_{n_{max}})/2$ led by the groundless intuition that it will render the smallest error as it is at equal distance from the two extremes of the design space, this will be suboptimal design, as the optimum S_n , depicted at Eq. (25), is notably below the midpoint. It is also worth to take into account that Eq. (26) represents a conservative upper bound of the maximum error that is going to be achieved by optimal design. This is because we have not considered the relation between $V'_1(t)$ and $V'_2(t)$ imposed by Eq. (15):

$$V'_1(t_{ext}) - V'_2(t_{ext}) = [V'_1(0) - V'_2(0)] e^{-\frac{2G_M}{C} t_{ext}} \quad (28)$$

This means that not all the possible values contained in $\triangle ABC$, defined on the $V'_{1_x} - V'_{2_x}$ plane, not $V'_1(0) - V'_2(0)$, will be covered by all the possible trajectories of the circuit. Thus by equating Eq. (19) to zero after having substituted the trajectories by their actual waveforms, t_{ext} is obtained:

$$t_{ext} = \frac{\tau}{2} \frac{\ln G_M R}{G_M R - 1} \quad (29)$$

and the exact value of the error in the extreme is then given by:

$$\varepsilon(t_{ext}) = \frac{V'_1(0) - V'_2(0)}{2} (1 - G_M R) (G_M R)^{-\left(\frac{G_M R}{G_M R - 1}\right)} \quad (30)$$

Unfortunately, to obtain the position of the extreme error in a closed form from this equation is not possible, but numerically we have confirmed that the S_n rendering the smallest error is the one depicted at Eq. (25).

3.2 Extrapolated results for larger networks

The results described above have been applied to the design of a 64×64 MOS-based resistive grid. Simulations have been realized using standard $0.35\mu\text{m}$ CMOS 3.3V process transistor models in HSPICE. The signal range at the nodes is $[0\text{V}, 1.5\text{V}]$, wide enough to evidence any excessive influence of the MOSFET nonlinearities in the spatial filtering. V_G is established at 3.3V. We aim to implement an array of capacitors interconnected by network of resistors with a time constant of $\tau = 100\text{ns}$.

Fig. 6 (a) Original image, (b) MOS-diffused image at instant of maximum error, (c) image diffused by resistor network, (d) absolute error, (e) absolute error multiplied by 10, (f) absolute error normalized to maximum individual pixel error

For that we will assume a resistor of $R = 100\text{k}\Omega$ and a capacitor of $C = 1\text{pF}$. S_n is obtained according to Eq. (25). But this equation does not take into account the substrate effect, or in other words, S_n is not only depending on the sum $V_{max} + V_{min}$ but also in the specific values of the initial voltages at drain and source that render the same sum. For a specific value of S_n , and the same $V'_1(0) + V'_2(0)$, the resistance

Fig. 7 RMSE of the MOS-based grid state vs. resistor grid state: (a) w/o mismatch, (b) Monte Carlo with 10% mismatch

implemented can vary $\pm 5\%$. We have selected $W = 0.4\mu\text{m}$ and $L = 7.54\mu\text{m}$ ¹, that result in an average resistance of $100\text{k}\Omega$ for all the possible initial conditions rendering the optimum sum, i. e. $V_{\max} + V_{\min}$.

The initial voltage at the capacitors is proportional to the image intensity displayed at Fig. 6(a). A MOS-based resistor network runs the diffusion of the initial voltages in parallel with an ideal linear resistor network. The deviation is measured via the RMSE (Fig. 7) and reaches a maximum soon after the beginning of the diffusion process. The state of the corresponding nodes in both networks at this point, displayed in Figs. 6(b) and 6(c), can be compared, Figs. 6(d), 6(e) and 6(f). The maximum observed RMSE for the complete image is 0.5% , while the maximum individual pixel error is 1.76% . This error remains below 0.6% even introducing an exaggerated mismatch (10%) in the transistors' V_{Tn0} and μ_n (Fig. 7(b))².

3.3 Diffusion duration control

Implementing a Gaussian filter, Eq. (13), by means of a dynamic diffusion requires a precise control of the diffusion duration, t , given that the scale, ξ , i. e. the square of the smoothing factor, is related with it through:

$$\xi = \frac{2t}{\tau} \quad (31)$$

Therefore, to filter the image with a Gaussian function of scale ξ means to let the diffusion run for $t = \tau\xi/2$ seconds. The actual value of τ is not important, as long as the necessary resolution of t for the required ξ is physically realizable. Actually, ξ is not determined by t itself, but by the quotient t/τ . In other words, a different selection of R and C will still render the same set of ξ 's as long as the necessary fraction of τ can be fine-tuned. Implementing this fine control of the diffusion duration is not a trivial task when we are dealing with τ 's in the hundred-nanosecond range. In order to provide robust sub- τ control of the diffusion duration, the operation must rely on internally, i. e. on-chip, generated pulses. Otherwise, propagation delays will render this task very difficult to accomplish.

We propose a method for the fine control of t based on an internal VCO. This method has been tested in the prototype that we are describing later in the chapter. The first block of the diffusion duration control circuit is the VCO (Fig. 8(a)). It consists on a ring of pseudo-NMOS inverters in which the load current is controlled by V_{bias} , thus modifying the propagation delay of each stage. As the inverter ring is composed by an odd number of stages, the circuit will oscillate. A pull-up transistor, with a weak aspect, have been introduced to avoid start up problems. Also a flip-flop

¹ This transistor length lies out of the physical design grid, that fixes the minimum feature size to be $0.05\mu\text{m}$. We are using it here as illustrative of the design procedure.

² Global deviations within the process parameters space have not been considered. In that case, the nominal resistance being implemented differs from the prescribed $100\text{k}\Omega$

Fig. 8 (a) 15-stage inverter ring VCO and (b) diffusion control logic

is placed to render 50% duty-cycle at the output. This circuit provides an internal clock that will be employed to time pulses that add up to the final diffusion duration.

The main block of the diffusion control is the 12-stage shift register. It will store a chain of 1's indicating how many clock cycles will the diffusion process run. The clock employed for this will be either external or the already described internal VCO³. The output signal, *diff_ctrl* in Fig. 8(b), is a pulse with the desired duration of the diffusion, that is delivered to the gates of the MOS-resistors.

3.4 Image energy computation

For real-time detection and tracking of dynamic textures we will be interested in a simplified representation of the scene. It can be constructed from the filtered image by first dividing it into subimages, usually of the same size. Each subimage is then represented by a number that encodes information related with the spatial frequency content of the subimage. This number is the image energy, as defined in Eq. (3). The energy of the image bin can be expressed as a function of time:

$$E_{kl}(t) = \sum_{\mathbf{k}} E_{kl}(\mathbf{k}; t) = \sum_{u=0}^{m-1} \sum_{v=0}^{n-1} |\hat{V}_{uv}(t)|^2 = \sum_{i=1}^m \sum_{j=1}^n |V_{ij}(t)|^2 \quad (32)$$

³ The aim of the internal VCO is to reach a better time resolution than an external clock. For programming the appropriate sequence into the SHR an external, and slower, clock is usually preferred

Fig. 9 In-pixel energy computing circuit

meaning that the energy of the image at time t accounts for the frequency components that have not been filtered yet. In terms of the dynamics of the resistor grid, the total charge in the array of capacitors is conserved but, naturally, the system evolves toward the less energetic configuration. Therefore the energy at time instant t indicates how the subimage has been affected by the diffusion until that exact point in time. The longer t the less of $E_{kl}(0)$ will remain in $E_{kl}(t)$. The energy lost between two consecutive points in time during the diffusion corresponds to that of the spatial frequencies filtered. Notice also that changing the reference level for the amplitude of the pixels does not have an effect beyond the dc component of the image spectrum. A constant value added to every pixel does not eliminate nor modify any of the spatial frequency components already present, apart from changing the dc component.

In order to analyze the presence of different spatial frequency components within a particular bin of the image, we would need to measure the energy of the bin pixels once filtered. Remember that for analyzing a particular band of frequencies we will subtract two lowpass filtered versions of the image. In this way, only the components of the targeted frequency band will remain. This will allow to track changes at that band without pixel-level analysis. The hardware employed to calculate the energy of the bins at the pixel-level (Fig. 9) consists in a transistor, M_E , that converts the pixel voltage to a current that is proportional to the square of the voltage, a switch S_E to control the amount of charge that will be subtracted from the capacitor C_E that realizes charge to voltage conversion. All the C_E 's within the bin will be connected to the same node. At the beginning, all of them are pre-charged to V_{REF} . Then, for a certain period of time, T_E , the transistor M_E is allowed to discharge C_E . But because the $m \times n$ capacitors of the bin are tied to the same node, the final voltage representing the bin energy after t seconds of diffusion will be:

$$V_{E_{kl}}(t) = V_{REF} - \frac{k_E S_E T_E}{mn C_E} \sum_{i=1}^m \sum_{j=1}^n [V_{ij}(t) - V_{T_{n0}}]^2 \quad (33)$$

We are assuming that all the M_E 's are nominally identical and operate in saturation. The offset introduced by $V_{T_{n0}}$ does not affect any spatial frequency other than the dc component. Deviations occur from pixel to pixel due to mismatch in the threshold voltage ($V_{T_{n0}}$), the transconductance parameter (k_E), and the body-effect constant (γ_E , not in this equation). Being area dependent effects, transistors M_E are

tailored to control the resulting error in the computation. Also, mobility degradation contributes to the deviation from the behaviour depicted in Eq. (33), what will ultimately limit the useful signal range.

4 Prototype texture segmentation chip

4.1 Chip description and data

The floorplan of the prototype chip is depicted in Fig. 10. It is comprised of a mixed-signal processing array, concurrent with the photosensor array, intended to carry out Gaussian filtering of the appropriate scale, within user-defined image areas. In addition, the total energy associated with each image bin is calculated. On the periphery, there are circuits for bias and control of the operation. The outcome of the processing can be read out pixel-by-pixel in a random access fashion. The value of the pixel is buffered at the column bus and delivered to either an analog output pad or an on-chip 8-bit SAR ADC.

The elementary cell of the analog core (Fig. 11) was described in [18]. It contains a diffusion network built with p-type MOS transistors. The limits of the diffusion areas are column-wise and row-wise selected enabled by the appropriate connection pattern. In this way, scale spaces can be generated at different user-defined areas of the scene. The pulse which determines the duration of the diffusion can be either

Fig. 10 Floorplan of the prototype chip

Fig. 11 Schematic of the elementary cell

externally input or internally generated, as mentioned in Sect. 3.3. The main characteristics of the chip are summarised in Table 1. A microphotograph of the prototype chip with a close-up of the photosensors is shown in Fig. 12.

<i>Technology</i>	0.35 μ m CMOS 2P4M
<i>Vendor (Process)</i>	Austria Microsystems (C35OPTO)
<i>Die size (with pads)</i>	7280.8 μ m $\times$ 5780.8 μ m
<i>Pixel size</i>	34.07 μ m $\times$ 29.13 μ m
<i>Fill factor</i>	6.45%
<i>Resolution</i>	QCIF: 176 $\times$ 144 px
<i>Photodiode type</i>	n-well/p-substrate
<i>Power supply</i>	3.3V
<i>Power consumption (including ADC)</i>	1.5mW
<i>Internal clock freq. range</i>	10-150MHz
<i>ADC throughput (I/O limit)</i>	0.11MSa/s (9 μ s/Sa)
<i>Exposure time range</i>	100 μ s-500ms

Table 1 Prototype chip data

Fig. 12 Microphotograph of the prototype chip

4.2 Linearity of the scale-space representation

As expected from simulations, the use of MOS transistors instead of true linear resistors in the diffusion network (Fig. 3(b)) achieves moderate accuracy even under strong mismatch conditions. However, the value of the resistance implemented by the transistors and therefore the value of the network time constant, τ , is quite sensitive to the process parameters. In order to have a precise estimation of the scale implemented by stopping the diffusion at different points in time —recall that $\xi = 2t/\tau$ — the actual τ needs to be measured. We have provided access to the extremes of the array and have characterized τ from the charge redistribution between two isolated pixels. The average τ measured is of 71.1ns ($\pm 1.8\%$). Attending to the technology corners, the predicted range for τ was $[49, 148]\text{ns}$. By reverse engineering the time constant, using Eq. (25), the best emulated resistance (R_{eq}) is obtained. Once τ is calibrated, any on-chip scale space can be compared to its ideal counterpart obtained by solving the spatially-discretized diffusion equation corresponding to a network consisting on the same C 's and resistors of value R_{eq} . In order to generate an on-chip scale space, a single image is captured. This image is converted to the digital domain and delivered through the output bus. It becomes the initial image of both the on-chip scale space and the ideal scale space calculated off-chip. The rest of the on-chip scale space is generated by applying successive diffusion steps to the original captured image. After every step, the image is converted to digital and delivered to the test instruments to be compared to the ideal image generated by MATLAB (Fig. 13). The duration of each diffusion step is internally configured

as sketched in Sect. 3.3. A total of 12 steps have been realized over the original captured image. Six of them are represented in Fig. 13 (first row) and compared to the ideal images (second row). The last row contains a pictorial representation of the error, normalized in each case to the highest measured error on individual pixels, which are 0%, 24.99%, 19.39%, 6.17%, 3.58% and 6.68%, respectively. It can be seen how FPN eventually becomes the dominant error at coarse scales. Keep in mind that this noise is present at the initial image of both scale spaces, but it is only added to each subsequent image of the on-chip scale space because of the readout mechanism. The key point here is that the error is kept under a reasonable level despite no FPN post-processing is carried out. This fact together with the efficiency of the focal-plane operation is crucial for artificial vision applications under strict power budgets. Two additional issues are worth to be mentioned. First of all, the accuracy of the processing predicted by simulation [17] is very close to that of the first images of the scale space, where fixed pattern noise is not dominant yet. Secondly, it has been confirmed for this and other scenes that the second major source of error in the scale space generation comes from uniform areas where the pixel values fall on the lowest extreme of the signal range. The reason is that the instantaneous resistance implemented by a transistor when its source and drain voltages coincides around the lowest extreme of the signal range presents the maximum possible deviation with respect to the equivalent resistance considered. The point is that, in such

Fig. 13 Comparative of scale spaces along the time. The first row corresponds to the on-chip scale space, the second one corresponds to the ideal scale space and finally the third one corresponds to their normalized difference

Fig. 14 Independent scale spaces within four subdivisions of the focal plane

a situation, the charge diffused between the nodes involved is very small, keeping the error moderate despite the large deviation in the resistance.

4.3 *Subsampling modalities*

Finally, as a glimpse into the possibilities of the prototype, independent scale spaces were generated within four subdivisions of the focal plane programmed into the chip (Fig. 14). Because of the flexible subdivision of the image, the chip can deliver from full-resolution digital images to different simplified representations of the scene which can be reprogrammed in real time according to the results of their processing. As an example of the image subsampling capabilities of the chip, three different schemes are shown in Fig. 15. The first one corresponds to the full-resolution representation of the scene—it has been taken in the lab, by displaying a real outdoor sequence in a flat-panel monitor. The second one represents the same scene

Fig. 15 Example of the image sampling capabilities of the prototype

after applying some binning. In the third picture, the region of interest (ROI), in the center, is at full resolution while the areas outside this region are binned together. The binning outside the ROI becomes progressively coarser. This represents a foveatized version of the scene in which the greater detail is only considered at the ROI, and then the grain increases as we reach further. From the computational point of view, this organization translates into important computing resources savings.

5 Conclusions

This chapter has presented a feasible alternative for focal-plane generation of a scale space. Its intention is to realize real-time dynamic textures detection and tracking. For that, focal-plane filtering with a resistor grid results in a very low-power implementation, while the appropriate image subdivision accommodated to the size of the targeted features also contributes to alleviate the computing load. A methodology for the design of the MOS-based resistor network is explained, leading to optimal design of the grid. Also, the means for a simplified representation of the scene are provided at the pixel level. These techniques have been applied to the design of a prototype smart CMOS imager. Some experimental results confirming the predicted behaviour are shown.

Acknowledgements This work is funded by CICE/JA and MICINN (Spain) through projects 2006-TIC-2352 and TEC2009-11812 respectively.

References

1. R. Nelson, R. Polana, CVGIP: Image Understanding **56**(1), 78 (1992)
2. D. Chetverikov, R. Péteri, in *International Conference on Computer Recognition Systems (CORES'05)* (Rydzyń Castle, Poland, 2005), pp. 17–26
3. T. Amiaz, S. Fazekas, D. Chetverikov, N. Kiryati, in *International Conference on Scale Space and Variational Methods in Computer Vision (SSVM'07)* (2007), pp. 848–859
4. I. Akyildiz, T. Melodia, K. Chowdhury, Computer Networks **51**(4), 921 (2007)
5. R. Péteri, M. Huskies, S. Fazekas. Dyntex: A comprehensive database of dynamic textures (2006). <http://www.cwi.nl/projects/dyntex/>
6. M. Ballerini, N. Cabibbo, R. Candelier, A. Cavagna, E. Cisbani, I. Giardina, A. Orlandi, G. Parisi, A. Procaccini, M. Viale, V. Zdravkovic, Animal Behaviour **76**(1), 201 (2008)
7. B. Jahne, H. Haußecker, P. Geißler, *Handbook of Computer Vision and Applications* (Academic Press, 1999), vol. 2, chap. 4
8. J. Babaud, A.P. Witkin, M. Baudin, R.O. Duda, IEEE Trans. Pattern Anal. Mach. Intell. **8**(1), 26 (1986)
9. T. Lindeberg, International Journal of Computer Vision **30**(2), 79 (1998)
10. D.G. Lowe, International Journal of Computer Vision **60**(2), 91 (2004)
11. L. Raffo, S. Sabatini, G. Bo, G. Bisio, IEEE Transactions on Neural Networks **9**(6), 1483 (1998)
12. E. Vittoz, X. Arreguit, Electronic Letters **29**(3), 297 (1993)
13. K. Bult, J. Geelen, IEEE Journal of Solid-State Circuits **27**(12), 1730 (1992)

14. L. Vadasz, IEEE Transactions on Electron Devices **13**(5), 459 (1966)
15. H. Kobayashi, J. White, A. Abidi, IEEE Journal of Solid-State Circuits **26**(5), 738 (1991)
16. K. Hui, B. Shi, IEEE Transactions on Circuits and Systems-I **46**(10), 1161 (1999)
17. J. Fernández-Berni, R. Carmona-Galán, in *European Conference on Circuit Theory and Design (ECCTD'09)* (Antalya, Turkey, 2009)
18. J. Fernández-Berni, R. Carmona-Galán, in *12th Int. W. Cellular Nanoscale Networks and Apps. (CNNA)* (Berkeley, CA, 2010), pp. 453–458