Fast IIR Isotropic 2D Complex Gabor Filters with Boundary Initialization

Alexandre Bernardino *Member, IEEE,* and José Santos-Victor *Member, IEEE*

***Abstract*— Gabor filters are widely applied in image analysis and computer vision applications. This paper describes a fast algorithm for isotropic complex Gabor filtering that outperforms existing implementations. The main computational improvement arises from the decomposition of Gabor filtering into more efficient Gaussian filtering and sinusoidal modulations. Appropriate filter initial conditions are derived to avoid boundary transients, without requiring explicit image border extension. Our proposal reduces up to 39% the number of required operations with respect to state-of-the-art approaches. A full C++ implementation of the method is publicly available.**

I. Introduction

Fast algorithms for Gabor convolution have been proposed in [1], [2], and take advantage of the separability of isotropic Gabor functions in the horizontal and vertical directions. In [1] a fast Gabor filter approximation is implemented with 3 pole Infinite Impulse Response (IIR) filters. These are general purpose filters, whose parameters can be arbitrarily selected. In [2] separable Finite Impulse Response (FIR) filters are applied in a multi-resolution pyramid to implement very optimized real Gabor filtering. Multi-resolution approaches exploit the particular relationships between Gabor filter parameters to sub-sample the original image and obtain computational improvements. However, this methodology is application specific because only particular sets of parameters can be implemented. Recent work on object representation and recognition [3], [4], [5], require image analysis with general purpose Gabor Filters tuned to arbitrary orientations, scales and frequencies.

We propose methods to reduce the computational cost of general purpose 2D isotropic complex Gabor Filters. The method involves rewriting the Gabor Filters as multiplications with complex exponentials and convolutions with Gaussian functions. The motivation for this decomposition consists in the fact that state-of-the-art Gaussian convolution is more efficient than Gabor convolution. We focus on the isotropic case, where vertical/horizontal separable implementations exist, but our method can also be applied to the anisotropic case. In fact, a separable implementation of anisotropic Gaussian filtering was recently proposed, consisting in two 1D convolutions performed in non-orthogonal directions [6]. We describe our approach in detail and show that allows up to 39% reduction in computational complexity, with respect to classic Gabor filtering [1]. Comparison with frequency domain FFT based methods and multi-resolution techniques is also provided.

An essential contribution of this paper is the derivation of appropriate filter state initialization. Without appropriate initial conditions, filter output will present spurious transient responses near the signal boundaries. These effects are undesired because they generate artificial responses corresponding to step edges in the boundary of the image. Extending the image boundaries is a common solution but increases the computational cost of the filtering operations. We derive formulas to compute the initial conditions for all filtering steps, thus avoiding explicit boundary extension.

Experiments compare the outputs of the proposed filtering technique with FIR filters whose coefficients are sampled from the true Gabor functions. Several combinations of parameters and image types are analyzed. In average, the relative approximation errors are about 3%.

The paper is organized as follows. In section II we review some of the underlying theory of Gabor Filtering. In section III we analyse the isotropic case, where efficient separable implementations have been developed in the past. Section IV is dedicated to discrete implementation of state-of-the-art Gaussian and Gabor filtering. In section V, we introduce the main idea of the proposed algorithm, and show how to perform Gabor filtering using only Gaussian convolutions and scalar multiplications, reducing the overall computational cost. In section VI we derive the appropriate boundary conditions to avoid undesirable transients or explicit boundary extension. Finally, in section VIII we present benchmarking results and performance analysis of the proposed methods, and conclude with some remarks and ideas for future work.

II. Gabor Filters

Gabor functions are defined by the multiplication of a complex exponential function (the carrier) and a Gaussian function (the envelope). Let (x, y) be the spatial coordinates and $\mathbf{w}_\sigma(x, y)$ be a two dimensional Gaussian envelope with scale $\sigma = (\sigma_1, \sigma_2, \theta)$. The standard deviations σ_1 and σ_2 are oriented along directions θ and $\theta + \pi/2$, respectively:

$$\mathbf{w}_\sigma(x, y) = \frac{1}{2\pi\sigma_1\sigma_2} e^{-\frac{(x\cos\theta + y\sin\theta)^2}{2\sigma_1^2} - \frac{(y\cos\theta - x\sin\theta)^2}{2\sigma_2^2}}$$

Let $\mathbf{c}_{\lambda,\theta}(x, y)$, be a complex exponential carrier representing a plane wave with wavelength λ and orientation θ:

$$\mathbf{c}_{\lambda,\theta}(x, y) = e^{i\frac{2\pi}{\lambda}(x\cos\theta + y\sin\theta)}$$

To simplify notation, we will drop the pixel coordinates (x, y) whenever they are not essential. A two dimensional Gabor function is, thus, written as: $\mathbf{g}_{\sigma,\lambda,\theta} = \mathbf{w}_\sigma \cdot \mathbf{c}_{\lambda,\theta}$.

Image analysis by convolution with Gabor functions has been extensively studied in the literature, and provides a means to estimate the oriented local frequency content of image

A. Bernardino and J. Santos-Victor are with the Instituto de Sistemas e Robótica, Instituto Superior Técnico, Technical University of Lisbon. Email: {alex,jasv}@isr.ist.utl.pt

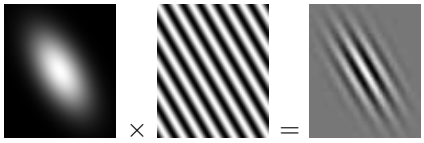

Fig. 1. A Gabor function (right) results from the product of a Gaussian envelope (left: $\alpha = 30$ degrees, $\sigma_1 = 8$ pixels, $\sigma_2 = 16$ pixels), by a complex exponential carrier (middle: $\lambda = 8$ pixels, $\theta = 30$ degrees). Only real parts are shown.

regions. In practical terms, the convolution output modulus will be high whenever the local image structure is similar to the Gabor function shape, in terms of scale (σ), wavelength (λ), and orientation (θ). Fig. 1 shows the real part of a two dimensional Gabor function, the corresponding Gaussian envelope $\mathbf{w}_\sigma$ and carrier $\mathbf{c}_{\lambda,\theta}$. The convolution of an image $\mathbf{f}$ with a Gabor $\mathbf{g}_{\sigma,\lambda,\theta}$ is written as: $\mathbf{z}_{\sigma,\lambda,\theta} = \mathbf{f} * \mathbf{g}_{\sigma,\lambda,\theta}$, and can be computed by the convolution integral:

$$\mathbf{z}_{\sigma,\lambda,\theta}(x,y) = \int_{x',y'} \mathbf{f}(x',y') \cdot \mathbf{g}_{\sigma,\lambda,\theta}(x-x', y-y') dx dy'$$

A. The Zero-Mean Gabor Filter

For image analysis, it is often desired to obtain invariance to image DC level offsets. In these cases, the zero-mean Gabor filter is used instead. To distinguish between the two classes of filters we name "classic Gabor filter" to the original form, and "zero-mean Gabor filter" to the zero-mean corrected case. We use the form given in [7]:

$$\mathbf{h}_{\sigma,\lambda,\theta} = \mathbf{w}_\sigma \cdot (\mathbf{c}_{\lambda,\theta} - \gamma_{\sigma,\lambda,\theta}) \tag{1}$$

The parameter $\gamma_{\sigma,\theta,\lambda}$ is set to remove the Gabor function DC value, i.e. such that its Fourier transform is zero at the origin, $\mathbf{H}(0,0) = 0$. Since the Gaussian window is normalized to have unit DC level, we obtain:

$$\gamma_{\sigma,\theta,\lambda} = \mathbf{W}_\sigma(-2\pi\cos\theta/\lambda, -2\pi\sin\theta/\lambda) \tag{2}$$

where $\mathbf{W}_\sigma$ denotes the Fourier transform of $\mathbf{w}_\sigma$.

The convolution of an image $\mathbf{f}$ with a zero-mean Gabor $\gamma_{\sigma,\lambda,\theta}$ is written as: $\bar{\mathbf{z}}_{\sigma,\lambda,\theta} = \mathbf{f} * \mathbf{h}_{\sigma,\lambda,\theta}$. Using the definition of the zero-mean Gabor filter (1), we get:

$$\bar{\mathbf{z}}_{\sigma,\lambda,\theta} = \mathbf{f} * \mathbf{g}_{\sigma,\lambda,\theta} - \gamma_{\sigma,\lambda,\theta}\mathbf{f} * \mathbf{w}_\sigma \tag{3}$$

Thus, image convolution with a zero-mean Gabor filter can be implemented by subtracting two terms: the convolution with a classic Gabor filter and a scaled convolution with a Gaussian filter. In practice it is convenient to perform the classic Gabor and Gaussian filters separately and subtract the correction term afterwards, instead of using directly zero-mean filter, because this filter is not separable.

III. The Isotropic Case

In the isotropic case, the Gaussian envelope is radially symmetric ($\sigma_1 = \sigma_2 = \sigma$), and the isotropic Gaussian function is defined in space and frequency by:

$$\mathbf{w}_\sigma(x,y) = \frac{1}{2\pi\sigma^2} e^{-\frac{x^2+y^2}{2\sigma^2}}, \quad \mathbf{W}_\sigma(\Omega_x,\Omega_y) = e^{-\frac{\sigma^2(\Omega_x^2+\Omega_y^2)}{2}}$$

The motivation to consider this case comes from the fact that Gaussian and Gabor Filters become separable in the x and y directions, and can be written as the tensor product of vertical and horizontal 1D filters. From now on we distinguish 1D and 2D functions by using, respectively, bold face and regular face fonts. Gaussian functions are decomposed in:

$$\mathbf{w}_\sigma(x,y) = \overbrace{\frac{1}{\sqrt{2\pi}\sigma} e^{-\frac{x^2}{2\sigma^2}}}^{w_\sigma(x)} \cdot \overbrace{\frac{1}{\sqrt{2\pi}\sigma} e^{-\frac{y^2}{2\sigma^2}}}^{w_\sigma(y)}$$

and Gabor functions are written as:

$$\mathbf{g}_{\sigma,\lambda,\theta}(x,y) = \underbrace{\frac{1}{\sqrt{2\pi}\sigma} e^{-\frac{x^2}{2\sigma^2}+i\frac{2\pi x\cos\theta}{\lambda}}}_{g_{\sigma,\lambda,\theta}(x)} \cdot \underbrace{\frac{1}{\sqrt{2\pi}\sigma} e^{-\frac{y^2}{2\sigma^2}+i\frac{2\pi y\sin\theta}{\lambda}}}_{g_{\sigma,\lambda,\theta}(y)}$$

Image convolution with such functions can be performed with two cascaded (horizontal and vertical) 1D convolutions. For example, Gaussian filtering can be implemented by:

$$\mathbf{w}_\sigma(x,y) * \mathbf{f}(x,y) = w_\sigma(y) * w_\sigma(x) * \mathbf{f}(x,y)$$

Since very fast one-dimensional IIR Gabor and Gaussian filters have been developed in the last decade [8], [9], [1], the isotropic case allows very efficient 2D implementations.

In this case, the zero-mean correction term (2), is independent of orientation: $\gamma_{\sigma,\lambda} = \exp\left(-2\sigma^2\pi^2/\lambda^2\right)$.

IV. The Discrete Case

In discrete coordinates, filters are designed to approximate the continuous operations both in space and in frequency. To date, the fastest implementations of convolution with discrete Gaussian and Gabor functions are described in [9] and [1]. In [9] a recursive separable Gaussian filter requires 7 real multiplications and 6 real additions per pixel per dimension. The extension to 2-dimensional signals thus requires 26 operations. In [1] a recursive separable classic Gabor filter, consisting in the cascade of a forward and a backward passes, implemented by equations: $v(n) = in(n) - \sum_{i=1}^{3} fd_i \cdot v(n-i)$ and $out(n) = B \cdot v(n) - \sum_{i=1}^{3} bd_i \cdot out(n-i)$, where fd_i and bd_i are, respectively, the coefficients of the forward and backward filters, B is a normalizing gain and v is the output of the forward pass. This implementation involves 1 real addition, 6 complex multiplications, 5 complex additions and 1 multiplication between a real and a complex number, adding to 49 operations. With 2-dimensional signals, this implementation requires 98 operations[1]. Image convolution with zero-mean Gabor filters consists in 1 Gaussian filtering, 1 Gabor filtering, 1 multiplication and one addition, totalizing 126 operations per pixel.

In this work we implement Gabor filters using Gaussian filtering operations. We adopt the 1D forward and backward 3 pole IIR filters from [9], defined by the Z transforms:

$$W_\sigma^f(z) = b_0/Q(z) \quad , \quad W_\sigma^b(z) = b_0/Q(z^{-1})$$

where $b_0 = 1 + a_1 + a_2 + a_3$ and:

$$Q(z) = 1 + a_1 z^{-1} + a_2 z^{-2} + a_3 z^{-3} \tag{4}$$

[1]we consider 1 complex mult equal to 4 real mult's plus 2 real add's.

Filter coefficients, a_1, a_2 and a_3 depend on filter scale (σ). Formulas to compute their values are provided in [9]. The full 1D filter is represented by $W_\sigma(z) = b_0^2/\left[Q(z)Q(z^{-1})\right]$ which, in the Fourier domain ($z = e^{i\omega}$), can be written as:

$$W_\sigma(e^{i\omega}) = b_0^2/(d_0 + d_1\cos(\omega) + d_2\cos(2\omega) + d_3\cos(3\omega))$$

with $d_0 = 1 + a_1^2 + a_2^2 + a_3^2$, $d_1 = 2(a_1 + a_1a_2 + a_2a_3)$, $d_2 = 2(a_2 + a_1a_3)$ and $d_3 = 2a_3$. Finally, the 2D discrete filter frequency representation is:

$$\mathbf{W}_\sigma(e^{i\omega_x}, e^{i\omega_y}) = W_\sigma(e^{i\omega_x})W_\sigma(e^{i\omega_y})$$

and the zero-mean correction parameter γ is computed using the discrete version of Eq. (2):

$$\gamma_{\sigma,\theta,\lambda} = \mathbf{W}_\sigma(e^{-i\frac{2\pi\cos\theta}{\lambda}}, e^{-i\frac{2\pi\sin\theta}{\lambda}})$$

V. Gabor Filtering Decomposition

This section presents the core of our approach. We propose a decomposition for Gabor filtering consisting in multiplications with complex exponentials and convolutions with Gaussian functions. The motivation for this decomposition is that state-of-the-art Gaussian filtering is more efficient than Gabor filtering, compensating the extra multiplications with complex exponentials. We consider the isotropic case, where vertical/horizontal separable implementations for Gaussian filtering exist, but the decomposition described here can also be applied to the anisotropic case.

We will provide a detailed analysis of the computational cost of the proposed method in comparison to implementations based on [1]. We focus our analysis in counting the operations that depend directly on the input signal. Variables not involving the input signals can be precomputed at initialization and stored for future use, e.g. the complex exponentials $c_{\lambda,\theta}$, the zero-mean scale factor $\gamma_{\sigma,\lambda,\theta}$ and the filter coefficients. Of course, this assumes that the filter parameters must be set *a priori* for each application.

Discrete convolution with a Gabor filter can be written as:

$$\mathbf{z}_{\sigma,\lambda,\theta}(x,y) = \sum_{k,l} \mathbf{f}(k,l)\cdot\mathbf{w}_\sigma(x-k,y-l)\cdot\mathbf{c}_{\lambda,\theta}(x-k,y-l)$$

Since the complex exponential function $\mathbf{c}_{\lambda,\theta}$ is separable, we can expand the previous expression into:

$$\mathbf{z}_{\sigma,\lambda,\theta}(x,y) = \mathbf{c}_{\lambda,\theta}(x,y)\cdot\sum_{k,l} \mathbf{c}^*_{\lambda,\theta}(k,l)\cdot\mathbf{f}(k,l)\cdot\mathbf{w}_\sigma(x-k,y-l)$$

where $\mathbf{c}^*$ denotes complex conjugation. In compact form:

$$\mathbf{z}_{\sigma,\lambda,\theta} = \mathbf{c}_{\lambda,\theta}\cdot\left[\left(\mathbf{f}\cdot\mathbf{c}^*_{\lambda,\theta}\right) * \mathbf{w}_\sigma\right] \quad (5)$$

Considering the isotropic case and adopting the IIR Gaussian filtering implementation of [9] (26 operations per pixel), the required computations on Eq. (5), are:

- A **modulation** (product of $\mathbf{f}$ with $\mathbf{c}^*_{\lambda,\theta}$) is computed by multiplying one real image and one complex image, corresponding to **2 operations** per pixel.
- A **complex Gaussian filtering** (convolution of $\mathbf{w}_\sigma$ with $\mathbf{f}\cdot\mathbf{c}^*_{\lambda,\theta}$) requires **52 operations** per pixel.

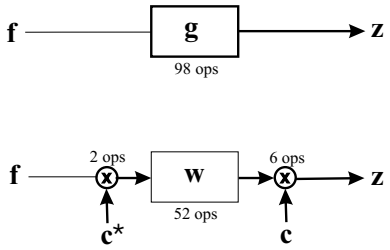

Fig. 2. 2D convolution with Gabor filters. Straightforward implementation if [1] require 98 operations per pixel (top), while the proposed equivalent decomposition method (bottom), only requires 60. Thick/Thin lines and boxes represent complex/real signals and filters, respectively.

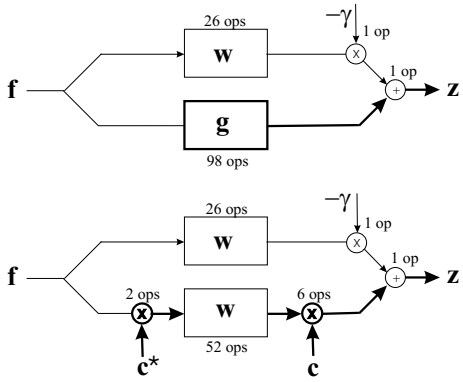

Fig. 3. 2D convolution with zero-mean Gabor filters, using classic Gabor and Gaussian filters, require 126 operations per pixel (top), while the proposed equivalent decomposition method (bottom), only requires 88. Thick/Thin lines and boxes represent complex/real signals and filters, respectively.

- A **demodulation** operation (product of $\left(\mathbf{f}\cdot\mathbf{c}^*_{\lambda,\theta}\right) * \mathbf{w}_\sigma$ with $\mathbf{c}_{\lambda,\theta}$) requires 1 complex multiplication per pixel, corresponding to **6 operations** per pixel.

Altogether we have 60 operations which, in comparison with the reference value of 98 operations, correspond to about 39% savings in computation. Fig. 2 illustrates the filtering process in a graphical form.

A. Zero-mean correction

Including the zero-mean correction term, the filtering operation (3) can be written as:

$$\bar{\mathbf{z}}_{\sigma,\lambda,\theta} = \mathbf{c}_{\lambda,\theta}\cdot\left[\left(\mathbf{f}\cdot\mathbf{c}^*_{\lambda,\theta}\right) * \mathbf{w}_\sigma\right] - \gamma_{\sigma,\lambda,\theta}\cdot(\mathbf{f} * \mathbf{w}_\sigma) \quad (6)$$

In this case we need the following additional operations:

- A **real Gaussian filtering** ($\mathbf{f} * \mathbf{w}_\sigma$) requiring **26 operations** per pixel.
- A **real scaling** by $\gamma_{\sigma,\lambda,\theta}$, requires **1 operation** per pixel.
- The **final subtraction**, corresponds to only **1 operation** per pixel because only the real part of Gabor functions have non zero DC value.

In total, zero-mean Gabor filtering requires 88 operations, corresponding to 30% savings when compared to the reference value of 126 operations. A graphical comparison of methods, for the zero-mean case, is presented in Fig. 3.

When multiple carriers (orientations/wavelengths) are considered, the term $\mathbf{f} * \mathbf{w}_\sigma$ in (6) is common to all of them. Thus, it may be computed only once and applied to all classic Gabor

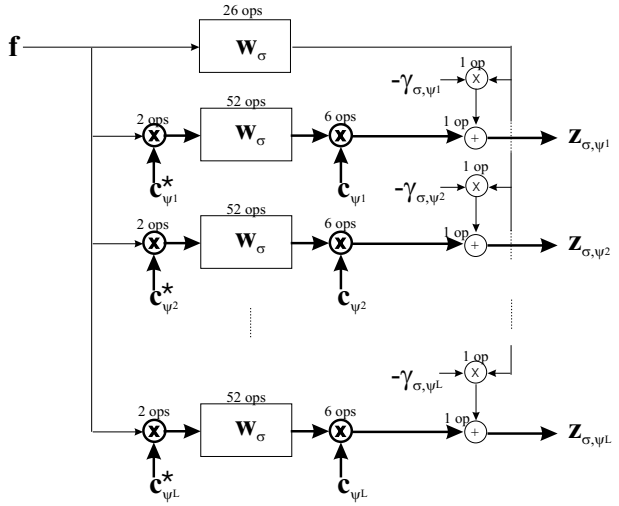

Fig. 4. 2D Zero-mean Gabor filtering with multiple carriers. Thick/Thin lines and boxes represent complex/real signals and filters, respectively. Close to each computational element we indicate the number of real operations required.

Image size	Operations per pixel
64x64	62.03
128x128	75.36
256x256	88.59
512x512	101.87

TABLE I
REAL OPERATIONS PER PIXEL IN THE FFT ALGORITHM OF [10].

filtering results at the same scale. A graphical representation of the method is shown in Fig. 4 for the multiple-carrier case.

B. Discussion

In this section we have presented a methodology for reducing the computational cost of general purpose Gabor filtering, implemented in the time domain by separable IIR filters. Fast linear space-invariant filtering techniques are also very frequently applied in the frequency domain, with the aid of the Fast Fourier Transform. It is thus convenient to compare our approach to FFT based ones. To date, on of the fastest implementations of the FFT is presented in [10]. The number of operations for the implementation of 2D filtering on real images is dependent on the size of the images and is shown in Table I for several image dimensions. From the table we observe that the method presented in this paper, in the classic Gabor filter case (60 ops), compare favorably to the FFT for image sizes larger that 64x64. Including zero-mean correction, our algorithm has a favorable operation count (88 ops) for images larger that 256x256. It must be taken into account that FFT methods can not compensate for boundary effects without explicitly extending image boundaries, thus requiring larger image sizes. In the following section we will fully derive the filter initialization conditions, for our proposal, without increasing image size. This fact presents an additional advantage over the FFT.

Another point worth discussing is the comparison with multi-resolution approaches. In [2] an approach for Gabor filtering uses a multi-resolution pyramidal implementation to compute the even Gabor decomposition of images into 4 different scales and orientations, with 190 operations per pixel. The same set of filters is applied to reduced versions of the image through low-pass filtering and down sampling in a factor of 2. In [11] a multi-resolution approach without sub-sampling, based on the *a trous* filter [12], requires 704 operations for a similar decomposition, but in this case even and odd Gabor responses are computed. To obtain a similar decomposition, our method would require 1408 operations per pixel. However we must stress that pyramidal implementations require particular settings or relationships between parameters, e.g. scales of the form 2^i, which limits the generality of the application. In this paper we are aiming at the general Gabor filtering problem, with unconstrained parameter settings.

Finally, we would like to comment on our evaluation of computational complexity. We are considering applications where Gabor Filter parameters can be defined at initialization. For example in video processing applications, banks of filters are defined at startup and then applied to all images in the sequence. In these cases, all variables depending only in the filtering parameters (and not on the images) can be computed off-line, e.g. the filter coefficients and the complex exponentials. Thus, our operation count only considered operations depending directly on the images to process. When filter parameters are required to change on-line, we incur in additional operations. In comparison with [1], since filter coefficients must be recomputed in both cases, the additional cost is concentrated in the computation of the complex exponentials for modulation/demodulation. Assuming that 1D complex exponentials can be efficiently computed by look-up tables, to obtain the 2D complex exponentials we will require one additional complex multiplication per pixel (6 ops). Therefore, our operation count would increase to 66 ops for classic Gabor filtering and 94 for zero-mean filters. Improvements would drop to 33% and 25% respectively.

VI. BOUNDARY CONDITIONS

An essential part of the proposed algorithm is the derivation of boundary conditions for recursive Gaussian filtering applied to modulated images. Without special initialization, filtering operations produce spurious transients at image boundaries. For complete transient extinction, the boundary should be extended by more than 3 times the scale of the filter. Thus, for large scale filters, this would imply a significant increase in computation time. A better solution is to derive adequate values to initialize the filter state at the boundaries, thus avoiding explicit filtering at the extended boundaries.

In the domain of signal processing, several approaches are common to address this problem, often involving the extension of the signal and making certain assumptions on signal properties, for instance constancy, continuous derivative or symmetry. Recently, [13] has developed a method for computing boundary conditions for IIR Gaussian filters when the original signal is extended by constancy. However, we must stress the fact that, in our method, the Gaussian filters are applied to images that result from the multiplication of

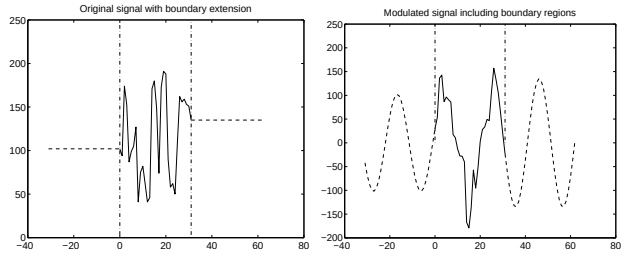

Fig. 5. Left: 1D signal boundary extension my constancy. Right: After modulation by a sinusoid.

Fig. 6. Left: Original image. Middle: Image with boundaries extended by constancy. Right: Extended image modulated by a complex exponential with parameters $\theta = 15$ degree and $\lambda = 29.6$ pixel.

complex exponentials with the original image. Therefore, the boundaries to deal with are not like any conventional boundary extension method proposed in the literature. This is illustrated in Fig. 5, for a 1D signal whose boundary has been extended by constancy. After modulation by a complex exponential, the boundary is not constant anymore, and can not be treated by conventional methods. The same applies for 2D signals. Fig. 6 shows an image whose boundary has been extended by constancy, before and after modulation by a sinusoid.

The 2D Gaussian filtering operation is applied to the modulated image in cascaded passes of horizontal forward (HF), horizontal backward (HB), vertical forward (VF) and vertical backward (VB) passes, as illustrated in Fig. 7.

Let $\mathbf{f}(x, y)$ denote the original image and $\mathbf{fm}$ be the

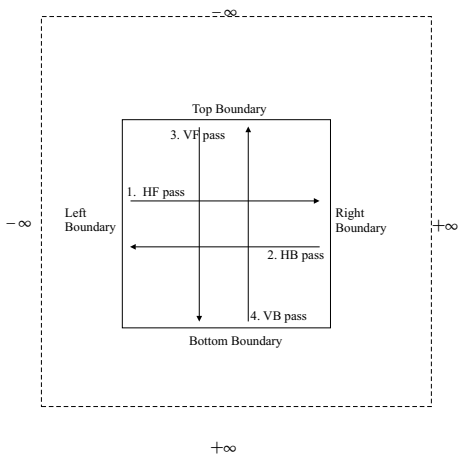

Fig. 7. Sequence of filtering operations, first in the horizontal directions (steps 1 and 2) and then in the vertical direction (steps 3 and 4). In boundary regions, it is considered that the original image, $\mathbf{f}(x, y)$, is extended by replicating the first and last values at each line/column in the adjacent boundary.

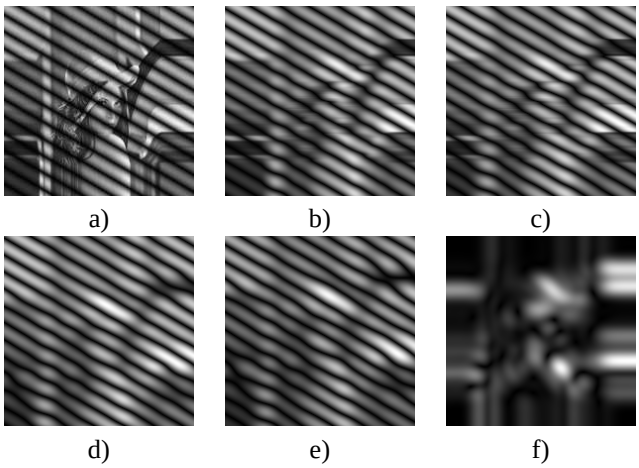

Fig. 8. Results of intermediate processing steps with explicit boundary extension. a) The modulated image: $\mathbf{fm}(x, y)$ b) After horizontal forward filtering: $\mathbf{hf}(x, y)$. c) After horizontal backward filtering: $\mathbf{hb}(x, y)$. d) After vertical forward filtering: $\mathbf{vf}(x, y)$. e) After vertical backward filtering: $\mathbf{vb}(x, y)$. f) After demodulation: $\mathbf{z}(x, y)$.

exponentially modulated image:

$$\mathbf{fm}(x, y; \omega_x, \omega_y) = \mathbf{f}(x, y) \cdot e^{i\cdot(\omega_x x + \omega_y y)}$$

where ω_x, ω_y are the horizontal and vertical frequencies of the complex exponential carrier, $\omega_x = 2\pi\cos(\theta)/\lambda$, $\omega_y = 2\pi\sin(\theta)/\lambda$. The full 2D Gaussian filter $\mathbf{w}_\sigma(x, y)$ is implemented separately by cascaded forward-backward 1D horizontal and vertical filters:

$$\mathbf{w}(x, y) = w^f(x) * w^b(x) * w^f(y) * w^b(y) = w(x) * w(y)$$

where subscript σ has been removed for convenience. The cascaded filtering operations generate the sequence of images:

$$\begin{cases} \mathbf{hf}(x, y) = w^f(x) * \mathbf{fm}(x, y), & \text{HF pass} \\ \mathbf{hb}(x, y) = w^b(x) * \mathbf{hf}(x, y), & \text{HB pass} \\ \mathbf{vf}(x, y) = w^f(y) * \mathbf{hb}(x, y), & \text{VF pass} \\ \mathbf{vb}(x, y) = w^b(y) * \mathbf{vf}(x, y), & \text{VB pass} \end{cases}$$

Intermediate results of the several processing steps are shown in in Fig. 8, including the extended boundary regions. At each pass, boundary regions change, not only as function of the image, but also as a function of the Gaussian and complex exponential parameters. Therefore, initial conditions must be computed explicitly for each pass.

A. Implicit Boundary Extension in Modulated Images

Horizontal Gaussian filtering operations are performed on image $\mathbf{fm}(x, y)$, whereas vertical Gaussian operations are applied to image $\mathbf{hb}(x, y)$. To compute the boundary conditions we need to represent the values of images $\mathbf{fm}(x, y)$ and $\mathbf{hb}(x, y)$ at the boundary regions before filtering.

Considering constancy in the boundary of the original image $\mathbf{f}(x, y)$, image $\mathbf{fm}(x, y)$ is computed by modulating the original image with complex exponentials and has the

following values in the boundary regions:

$$\begin{cases} \mathbf{fm}^l(x,y) = \mathbf{f}(0,y) \cdot e^{i(\omega_x x + \omega_y y)}, & x < 0 \\ \mathbf{fm}^t(x,y) = \mathbf{f}(x,0) \cdot e^{i(\omega_x x + \omega_y y)}, & y < 0 \\ \mathbf{fm}^r(x,y) = \mathbf{f}(0,N-1) \cdot e^{i(\omega_x x + \omega_y y)}, & x \geq N \\ \mathbf{fm}^b(x,y) = \mathbf{f}(N-1,0) \cdot e^{i(\omega_x x + \omega_y y)}, & y \geq N \end{cases} \quad (7)$$

where the superscripts "*l*", "*t*", "*r*" and "*b*" stand for the left, top, right and bottom boundary regions, respectively, and N is the image size (both horizontally and vertically). Image $\mathbf{hb}(x,y)$ is obtained from $\mathbf{fm}(x,y)$ by gaussian filtering and its top and bottom boundaries can be computed by $\mathbf{hb}(x,y) = \mathbf{fm}(x,y) * w(x)$. Using (7), the previous expression can be particularized for the top boundary region:

$$\begin{aligned} \mathbf{hb}^t(x,y) &= \sum_{x'} \mathbf{f}(x',0) e^{i(\omega_x x' + \omega_y y)} w(x-x') = \\ &= \left(\mathbf{fm}^t(x,0) * w(x)\right) e^{i\omega_y y} = \mathbf{hb}(x,0) \cdot e^{i\omega_y y} \end{aligned} \quad (8)$$

Analogoulsy, for the bottom boundary we have:

$$\mathbf{hb}^b(x,y) = \mathbf{hb}(x,0) \cdot e^{i\omega_y y} \quad (9)$$

To simplify notation we will consider, at each step, a single line or column in the image, and define the following signals:

- the input signal to the Gaussian filtering

$$in(t) = \begin{cases} \mathbf{fm}(t,y) & \text{horizontal pass} \\ \mathbf{hb}(x,t) & \text{vertical pass} \end{cases}$$

- the output of the forward filter

$$v(t) = \begin{cases} \mathbf{hf}(t,y) & \text{horizontal pass} \\ \mathbf{vf}(x,t) & \text{vertical pass} \end{cases}$$

- the output of the backward filter

$$out(t) = \begin{cases} \mathbf{hb}(t,y) & \text{horizontal pass} \\ \mathbf{vb}(x,t) & \text{vertical pass} \end{cases}$$

Additionally, each signal is break into three parts: (i) the left or top boundary; (ii) inside the image; and (iii) the right or bottom boundary. Subscripts "-" and "+" are used to indicate, respectively, cases (i) and (iii), e.g.:

$$in_-(t) = in(t), \quad t < 0 \quad \text{and} \quad in_+(t) = in(t), \quad t \geq N$$

Since the utilized filters are 3rd order, for each filtering stage we require 3 values at the boundary to initialize the filters. Let us consider that both image coordinates start at 0 (first pixel) and end at $N-1$ (last pixel). Therefore, we need to compute $v_-(t), t \in \{-1,-2,-3\}$ to initialize the forward pass, and $out_+(t), t \in \{N, N+1, N+2\}$ to initialize the backward pass.

B. Forward Boundary Conditions

Boundaries of the input signals $in(t)$ are given directly by the values at the boundaries of images **fm** and **hb**, in Eqs. (7-9). For horizontal filtering we have:

$$in_-(t) = \mathbf{f}(0,y) e^{i\omega_y y} \cdot e^{i\omega_x t} \quad (10)$$

$$in_+(t) = \mathbf{f}(N-1,y) e^{i\omega_y y} \cdot e^{i\omega_x t} \quad (11)$$

whereas for vertical filtering, input signal's boundaries are:

$$in_-(t) = \mathbf{hb}(x,0) \cdot e^{i\omega_y t} \quad (12)$$

$$in_+(t) = \mathbf{hb}(x,N-1) \cdot e^{i\omega_y t} \quad (13)$$

Thus, input signals at the left and top boundary regions are constituted by pure frequencies along the corresponding filtering direction. Therefore, to compute the forward filtering output at these boundaries, we just need to multiply by the frequency response of the forward filters. Thus, we have, for the horizontal stage:

$$v_-(t) = in_-(t) * w^f(t) = \mathbf{f}(0,y) e^{i\omega_y y} W^f(e^{i\omega_x}) \cdot e^{i\omega_x t}$$

and for the vertical stage:

$$v_-(t) = \mathbf{hb}(x,0) W^f(e^{i\omega_y}) \cdot e^{i\omega_y t}$$

Evaluating $v_-(-1)$, $v_-(-2)$ and $v_-(-3)$, we get the initial conditions for the forward passes. Since the complex exponentials and filter frequency responses do not depend on the input signal, their values can be precomputed and premultiplied offline, thus the run time cost for computing the forward boundary conditions is of 3 real multiplications per line and 3 complex multiplications per column.

C. Backward Boundary Conditions

To derive boundary conditions for the backward passes, we consider that the forward pass continues from the right or bottom boundaries to infinity, with input in_+ and output v_+, and compute the Z-transform of the result. Then, the backward pass starts at infinity with input v_+, output out_+ and zero initial conditions. The full Z-transform of this forward-backward simulated filtering at the boundary, along with the boundary values at the end of the forward image pass, will allow the determination of the initial conditions for image backward filtering: $out_+(N)$, $out_+(N+1)$ and $out_+(N+2)$.

To simplify notation we will shift the origin of coordinates to the right or bottom boundaries ($\tau = t - N, \tau \geq 0$). With this notation, the horizontal input signal (11) is given by:

$$in_+(\tau) = \mathbf{f}(N-1,y) e^{i(\omega_y y + \omega_x N)} \cdot e^{i\omega_x \tau}$$

and, in the vertical filtering stage, from (13):

$$in_+(\tau) = \mathbf{hb}(x,N-1) e^{i\omega_y N} \cdot e^{i\omega_y \tau}$$

Forward filtering now continues beyond the boundary and is represented by the recursive difference equation:

$$v_+(\tau) = b_0 in_+(\tau) - a_1 v_+(\tau-1) - a_2 v_+(\tau-2) - a_3 v_+(\tau-3)$$

with initial conditions $v_+(\tau) = v(t = N - \tau), \tau \in \{1,2,3\}$. Converting the previous difference equation to the unilateral Z transform domain, we get:

$$\begin{aligned} V_+(z) = \frac{b_0}{Q(z)} In_+(z) - v_+(-1)\frac{a_1 + a_2 z^{-1} + a_3 z^{-2}}{Q(z)} - \\ - v_+(-2)\frac{a_2 + a_3 z^{-1}}{Q(z)} - v_+(-3)\frac{a_3}{Q(z)} \end{aligned} \quad (14)$$

with $Q(z)$ as defined in (4). In the backward pass, $out_+(\tau)$, is computed by the anti-causal recursive difference equation:

$$out_+(\tau, y) = b_0 v_+(\tau) - a_1 out_+(\tau+1) - \\ - a_2 out_+(\tau+2) - a_3 out_+(\tau+3)$$

Because this filtering operations starts at ∞, we consider zero initial conditions[2]. In the Z-transform domain:

$$Out_+(z) = b_0 \frac{V_+(z)}{Q(z^{-1})} \tag{15}$$

Joining (14) and (15), we get out_+ as a function of the input signal, in_+, and the initial conditions in v_+:

$$Out_+(z) = \frac{b_0^2}{P(z)} In_+(z) - v_+(-1) b_0 \frac{a_1 + a_2 z^{-1} + a_3 z^{-2}}{P(z)} \\ - v_+(-2) b_0 \frac{a_2 + a_3 z^{-1}}{P(z)} - b_0 v_+(-3) \frac{a_3}{P(z)}$$

where $P(z) = Q(z)Q(z^{-1})$. To obtain the solution for $out_+(\tau)$ we must invert the previous Z transform. Details are given in appendix. The final solution is of the form:

$$out_+(\tau) = \sum_{j=1}^{3} v_+(-j)\alpha_j(\tau) + in_+(0) \cdot \beta(\tau)$$

Initial conditions for the backward image filtering step are now given by evaluating $out_+(0)$, $out_+(1)$ and $out_+(2)$. Since $\alpha_j(\tau)$ and $\beta(\tau)$ are complex coefficients than do not depend on the input image, their values can be computed at initialization. Therefore, the run time cost of computing the backward initial conditions consists in 12 complex multiplications and 3 complex additions per image line and column.

The full cost of computing the boundary conditions is obtained by summing the cost of the forward pass (3 real multiplication per line and 3 complex multiplications per column) to the cost of the backward pass (12 complex multiplications and 3 complex additions per image line and column). With square images of size N in both dimensions, this correspond to a number of $201 \times N$ real operations per image, i.e. $201/N$ operations per pixel. For example, with images of size 128×128 it corresponds to about 2 operations per pixel, and an efficiency penalty of about 2%. For larger image sizes, the efficiency loss becomes negligible.

VII. Results

To evaluate the quality of the developed method, we have performed several experiments comparing the proposed filtering implementation *versus* image convolution with FIR filters sampled from the continuous kernels. The quality index is defined as the average signal-to-error ratio on image sets. Let $I^n_{FIR}(x, y)$ be the result of convolving image n with FIR filters with image boundary extension, and let $I^n_{OUR}(x, y)$ be the result of image filtering with our method. Then, in a set of N images, the quality index is defined as:

$$SER\,[dB] = -\sum_{n=1}^{N} 10 \log_{10} \frac{\sum_{(x,y)} |I^n_{OUR}(x,y) - I^n_{FIR}(x,y)|^2}{\sum_{(x,y)} |I^n_{OUR}(x,y)^2|}$$

[2]In fact initial conditions are indetermined but finite, thus any transient vanishes before reaching the boundary

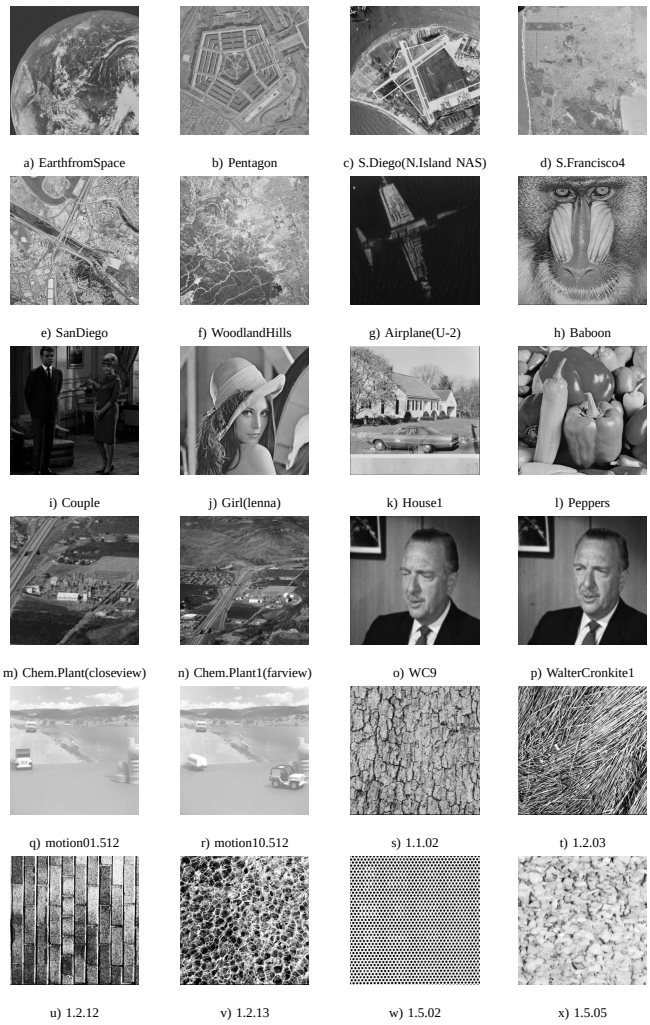

Fig. 9. Images chosen from the USC-SIPI database. Aerial images: a) to f). Miscellaneous: g) to l). Sequences: m) to r). Textures: s) to x)

We have used images from the USC-SIPI database [14]. This image dataset is composed by images from four different classes: aerial, miscellaneous, sequences and textures. We have chosen 6 images from each class, the ones presented in Fig 9.

Since our method is based on Gaussian IIR filters, we start by evaluating the approximation quality of those filters as a function of the scale parameter. Although this is not a contribution of this work, it will allow us to determine the scale range for which it is expected the method to perform well. The quality index has been computed for a large set of scales in all chosen images. In Fig. 10 we show the quality index for each scale and data set. Notice that the approximation is good (above 40dB, error = 1%) for the range of scales $2^{0.5} = 1.41$ to $2^{5.5} = 45.25$. Above $2^{5.5}$ the quality degrades abruptly. Therefore, in the following experiments, we will evaluate performance only for scales with good Gaussian filtering approximation (we use dyadic scales between $\sigma = 2$ and $\sigma = 32$). For scales lower than $\sigma = 2$, FIR filters are more efficient alternatives to IIR filtering because of their small size.

To evaluate the proposed Gabor filtering approximation we will consider the influence of all three parameters: scale, ori-

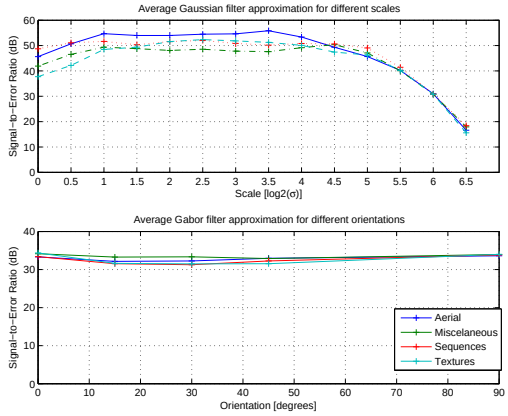

Fig. 10. Top: Average approximation quality of IIR Gaussian filters as a function of scale. Bottom: Average approximation quality of proposed Gabor filters as a function of orientation.

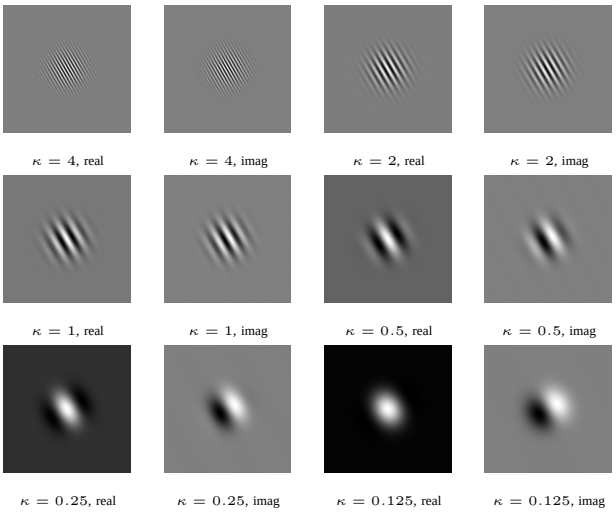

Fig. 11. Shape of Gabor functions with different wave number $\kappa = \sigma/\lambda$. Both real and imaginary parts are shown.

entation and wavelength. Experimentally we have verified that, in our dataset, approximation quality is not very sensitive to orientation. We have considered values $\theta \in \{0, 15, 30, 45, 90\}$ degrees, which, due to symmetry considerations, are good representatives of the whole set of orientations. Then, averaging the approximation errors over σ and λ we have obtained the results shown in Fig. 10, which illustrate that the influence of orientation is not critical in the approximation error.

To evaluate the influence of σ and λ we compute the approximation error for parameters $\sigma \in \{2, 4, 8, 16, 32\}$ and $\lambda \in \{2, 4, 8, 16, 32, 64\}$. To reduce computation times, only one orientation was considered: $\theta = 30$ degrees. Figs. 12 and 13, present the approximation quality for, respectively, the real and imaginary parts of Gabor filters, as a function of λ, for all considered scales and image classes. In practice some combinations of parameters may not be interesting to use. For instance, large scales and wavelengths may be obtained by first reducing the image resolution and applying smaller scale and wavelength filters. Also, very small wave numbers ($\kappa = \sigma/\lambda$) have large overlap in the frequency domain and

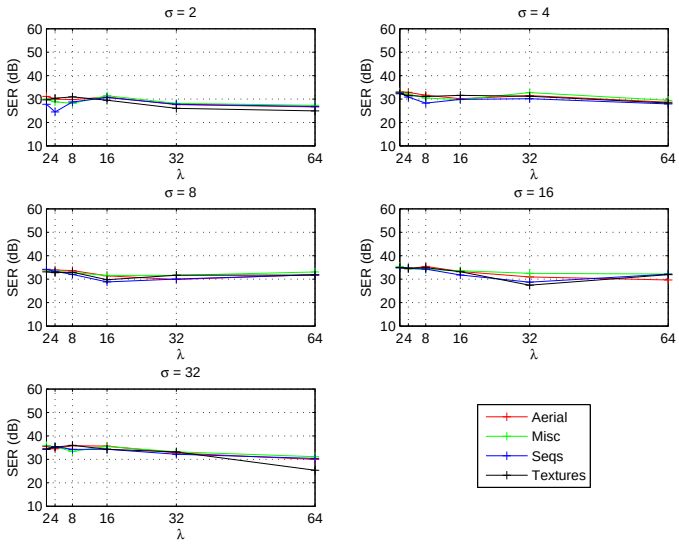

Fig. 12. Approximation quality of the Gabor imaginary part as a function of σ, λ and image type.

produce very similar outputs. In Fig. 11 we show the shape of Gabor filters for different wave numbers κ, and it is visible that for $\kappa < 0.25$, their shape do not change significantly, therefore no important additional information is extracted from the images. Notwithstanding, some application may require either to compute large scale and wavelength information in full resolution images, or use very small wave numbers, thus, for the sake of completeness, we present the results obtained for the whole set of parameters. For the imaginary part we can observe in Fig. 12 that performance is about 30dB (3% relative error) in average, and above 24dB (6% relative error) for the whole range of parameters. For the real part, average performance is also around 30dB but perfomance drops to about 20dB (10% relative error) for some images when $\sigma = 2$ and $\lambda = 4$, as can be seen in Fig. 13. The reason for this fact is a mismatch between the DC values of IIR and FIR filters, that happens to be larger at this range.

Different classes of images do not show significant differences in approximations quality. Anyway, images in the texture set have a bit lower approximation quality, especially for low values of the wave number, which correspond to frequency bands with low energy in textured images. The filter output at these bands has low amplitude, which decreases the approximation quality.

Summarizing, the proposed filters have a good fidelity for all classes of images, with the considered range of parameters: $\sigma \in [2, 32]$, $\lambda \in [2, 64]$, $\theta \in [0, 360]$. Scales smaller than 2 can be efficiently implemented by FIR filters instead. Large scale values with large wavelengths can be implemented in a multi-scale framework, where image resolution is first reduced to match the scale and wavelength range of existing filters. This is not possible, however, with small wavelengths, due to aliasing effects. Even though we have not tested scales higher that 32, the approximation error for small wavelengths do not decay significantly with scale, and it is likely that larger scales still have good approximations.

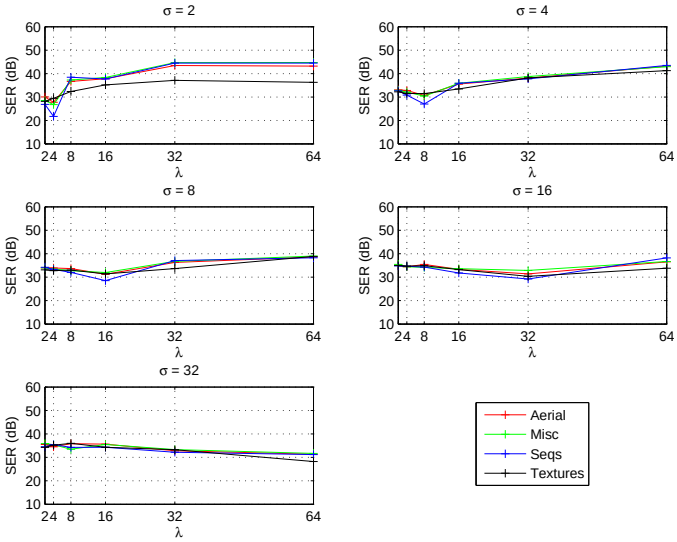

Fig. 13. Approximation quality of the Gabor real part with zero-mean correction as a function of σ, λ and image type.

VIII. Conclusions

We have presented a novel algorithm for the computation of Gabor features. Computational improvements with regard to state of the art methods are obtained by an efficient decomposition of Gabor convolution into Gaussian convolutions. The method is able to reduce computations up to 39%, when compared to other fast methods. Since the method is based on IIR filters, we provide the full derivation of the boundary conditions for filter initialization. Without such conditions, explicit image boundary extension would be required, penalizing the computational efficiency of the algorithm.

We have presented a quantitative evaluation of the approximation error between the proposed filters and FIR masks obtained by sampling the Gabor function, for a wide range of images and parameters. Results show that the approximation error is, in average, below 3%, which confirms the effectiveness of the approach. The computational analysis presented in this work assumes that Gabor filter parameters can be defined at initialization and remain constant during the duration of the application. This allows the pre-computation of several auxiliary variables dependent on the parameters. However, if the application requires arbitrary parameter change in runtime, computational penalties are incurred. Future work should address this issue and consider the cost of computing the auxiliary variables.

Several applications of Gabor analysis can be found nowadays in the literature, e.g. object representation [5], texture classification [15], [16], image segmentation [17], motion estimation [18], image compression [19] and visual attention [20]. The results presented in this paper thus have a wide spectrum of application, and may drive further research on several Gabor analysis related topics. A full C++ implementation of the method presented in this work is publicly available at the author's web page. Each filtering operation takes about 4ms on 128 by 128 images, with a P4 2.66 GHz computer.

Acknowledgements

This work has been partially supported by EU Proj. (IST-004370) RobotCub - ROBotic Open-Architecture Technology for Cognition, Understanding and Behavior.

References

[1] I. Young, L. van Vliet, and M. van Ginkel, "Recursive gabor filtering," *IEEE Trans. on Signal Processing*, vol. 50, no. 11, pp. 2798–2805, 2002.
[2] O. Nestares, R. Navarro, and J. Portilla, "Efficient spatial-domain implementation of a multiscale image representation based on gabor functions," *Journal of Electronic Imaging*, vol. 7, no. 1, pp. 166–173, Jan. 1998.
[3] P. Moreno, A. Bernardino, and J. Santos-Victor, "Gabor parameter selection for local feature detection," in *Proc. of the 2nd IBPRIA*, Estoril, Portugal, June 2005.
[4] B. Gokberk, L. Akarun, and E. Alpaydin, "Feature selection for pose invariant face recognition," in *Proc. of the 16th ICPR*, vol. vol.4, 2002, pp. 306–309.
[5] V. Krueger and G. Sommer, "Gabor wavelet networks for object representation," in *DAGM Symposium*, Kiel, Germany, September 2000.
[6] J.-M. Geusebroek, A. W. M. Smeulders, and J. van de Weijer, "Fast anisotropic gauss filtering," in *ECCV (1)*, 2002, pp. 99–112.
[7] T. Lee, "Image representation using 2d gabor wavelets," *IEEE Trans. on PAMI*, vol. 18, no. 10, October 1996.
[8] I. Young and L. van Vliet, "Recursive implementation of the gaussian filter," *Signal Processing*, vol. 44, pp. 139–151, 1995.
[9] L. van Vliet, I. Young, and P. Verbeek, "Recursive gaussian derivative filters," in *Proc. of the 14th ICPR*, Brisbane, Australia, 1998, pp. 509–514.
[10] B. Triggs and M. Sdika, "A modified split radix fft with fewer arithmetic operations," mid 2006, to appear in IEEE Trans. on Signal Processing. [Online]. Available: http://www.fftw.org/
[11] A. Bernardino and J. Santos-Victor, "A real-time gabor primal sketch for visual attention," in *Proc. of the 2nd IBPRIA*, Estoril, Portugal, June 2005.
[12] S. Mallat, *A Wavelet Tour of Signal Processing, 2nd Ed.* Academic Press, 1999.
[13] B. Triggs and M. Sdika, "Boundary conditions for young - van vliet recursive filtering," late 2005, to appear in IEEE Trans. on Signal Processing. [Online]. Available: http://lear.inrialpes.fr/pubs/2005/TS05
[14] S. University of Southern California and I. P. Institute, "The usc-sipi image database," http://sipi.usc.edu/services/database.
[15] P. Kruizinga, N. Petkov, and S. Grigorescu, "Comparison of texture features based on gabor filters," in *Proc. of the 10th ICIAP*, Venice, Italy, Sept. 1999, pp. 142–147.
[16] T. Randen and Husøy, "Image representation using 2d gabor wavelets," *IEEE Trans. on PAMI*, vol. 21, no. 4, pp. 291–310, April 1999.
[17] T. Tangsukson and J. Havlicek, "Am-fm image segmentation," in *Proc. IEEE ICIP*, Vancouver, Canada, Sept. 2000, pp. 104–107.
[18] E. Bruno and D. Pellerin, "Robust motion estimation using gabor spatial filters," in *Proc. of the 10th ESPC*, Sept. 2000.
[19] S. Fischer and G. Cristóbal, "Minimum entropy transform using gabor wavelets for image compression," in *Proc. of ICIAP*, Palermo, Italy, Sept. 2001.
[20] L. Itti and C. Koch, "A saliency-based search mechanism for overt and covert shifts of attention," *Vision Research*, vol. 40, pp. 1489–1506, 2000.

Appendix

In this appendix we invert the Z transform required for computing the backward boundary conditions in the filtering process of Section VI:

$$Out_+(z) = In_+(z)A(z) - v_+(-1)B(z) - v_+(-2)C(z) - v_+(-3)D(z)$$

where $A(z)$, $B(z)$, $C(z)$ and $D(z)$ are Z-transforms that only depend on filter parameters: $A(z) = b_0^2/P(z)$, $B(z) = b_0\left(a_1 + a_2 z^{-1} + a_3 z^{-2}\right)/P(z)$, $C(z) = b_0\left(a_2 + a_3 z^{-1}\right)/P(z)$, $D(z) = b_0 a_3/P(z)$. We decompose

the signal on a natural term $N(z)$ depending only on the initial conditions and a forced term $F(z)$ depending only on the input signal:

$$\begin{aligned} N(z) &= -v_+(-1)B(z) - v_+(-2)C(z) - v_+(-3)D(z) \\ F(z) &= In_+(z)A(z) \end{aligned}$$

To compute the time domain signals $n(t)$ and $f(t)$ we will perform a partial fraction expansion in first order terms. Let p_1, p_2 and p_3 be the poles of $Q(z)$. Then, $P(z) = Q(z)Q(z^{-1})$ can be written as:

$$P(z) = \prod_{i=1}^{3} \left(1 - p_i z^{-1}\right) \prod_{i=1}^{3} \left(1 - p_i z\right)$$

and its inverse is given by:

$$\frac{1}{P(z)} = \frac{b_0^2 a_3^{-1} z^{-3}}{\prod_{i=1}^{3} \left(1 - p_i z^{-1}\right) \prod_{i=1}^{3} \left(1 - p_i^{-1} z^{-1}\right)}$$

Performing a partial fraction expansion of the previous expression, we obtain:

$$\frac{1}{P(z)} = \sum_{i=1}^{3} \frac{R_i}{1 - p_i z^{-1}} + \sum_{i=1}^{3} \frac{R_i'}{1 - p_i^{-1} z^{-1}}$$

where R_i and R_i' are the residues of the causal and anti-causal terms respectively. To initialize the backward passes we only require values of $out_+(\tau)$ for $\tau \geq 0$. Therefore, only the causal residues need to be computed explicitly[3]:

$$R_i = \frac{1}{P(z)} \left(1 - p_i z^{-1}\right)|_{z=p_i}, \quad i = \{1, 2, 3\}$$

Now, the natural term $N(z)$ can be written as:

$$N(z) = -\sum_{i=1}^{3} \sum_{j=1}^{3} v_+(-j) \left(\frac{r(i,j)}{1 - p_i z^{-1}} + \frac{r'(i,j)}{1 - p_i^{-1} z^{-1}} \right)$$

with the natural causal residues $r(i, \tau)$ given by:

$$\begin{cases} r(i,1) = R_1 b_0 \left(a_1 + a_2 p_i^{-1} + a_3 p_i^{-2}\right) \\ r(i,2) = R_2 b_0 \left(a_2 + a_3 p_i^{-1}\right) \\ r(i,3) = R_3 b_0 a_3 \end{cases}$$

To obtain the time function $n(\tau),\ t \geq 0$, only the causal residues are used:

$$n(\tau) = \sum_{j=1}^{3} v_+(-j) \sum_{i=1}^{3} r(i,j) p_i^{\tau}, \quad \tau \geq 0$$

We now consider the forced response. Boundary input signals, (11) and (13), are of the form: $in_+(\tau) = in_+(0) p_4^{\tau}$. The values of $in_+(0)$ and p_4 depend on the processing step. In the horizontal filtering stage we have, from (11):

$$in_+(0) = \mathbf{f}(N-1, y) e^{i(\omega_y y + \omega_x N)} \quad \text{and} \quad p_4 = e^{i\omega_x}$$

and, in the vertical filtering stage, from (13):

$$in_+(0) = \mathbf{hb}(x, N-1) e^{i\omega_y N} \quad \text{and} \quad p_4 = e^{i\omega_y}$$

[3]the response associated to non-causal terms only exists for $\tau < 0$.

In the Z transform domain, the input signals are:

$$In_+(z) = in_+(0) \frac{1}{1 - p_4 z^{-1}}$$

The partial fraction expansion of $F(z)$ now involves the pole of the input signal.

$$F(z) = in_+(0) \cdot \left(\frac{r_4}{1 - p_4 z^{-1}} + \sum_{i=1}^{3} \frac{r_i}{1 - p_i z^{-1}} + \frac{r_i'}{1 - p_i^{-1} z^{-1}} \right)$$

where the forced response causal residues are given by:

$$r_i = R_i b_0^2 / \left(1 - p_4 p_i^{-1}\right), i \in \{1, 2, 3\}, \quad r_4 = b_0^2 / P(z)|_{z=p_4}$$

Consequently, the forced response $f(\tau)$ for $\tau \geq 0$ is given by:

$$f(\tau) = in_+(0) \cdot \sum_{i=1}^{4} r_i p_i^{\tau}$$

Finally, the desired initial conditions are obtained by adding the natural and forced terms, evaluated at $\tau = 0, 1, 2$:

$$out_+(\tau) = \sum_{j=1}^{3} v_+(-j) \cdot \underbrace{\sum_{i=1}^{3} r(i,j) p_i^{\tau}}_{\alpha_j(\tau)} + in_+(0) \cdot \underbrace{\sum_{i=1}^{4} r_i p_i^{\tau}}_{\beta(\tau)}$$

Complex coefficients $\alpha_j(\tau)$ and $\beta(\tau)$ only depend on the filter parameters and can be precomputed at initialization.

PLACE PHOTO HERE

Alexandre Bernardino received the PhD degree in Electrical and Computer Engineering in 2004 from Instituto Superior Tcnico (IST). He is an Assistant Professor at IST and Researcher at the Institute for Systems and Robotics (ISR-Lisboa) in the Computer Vision Laboratory (VisLab). He participates in several national and international research projects in the fields of robotics, cognitive systems, computer vision and surveillance. He published several articles in international journals and conferences, and his main research interests focus on the application of computer vision, cognitive science and control theory to advanced robotic and surveillance systems. Prof. Alexandre Bernardino is an IEEE Member.

PLACE PHOTO HERE

José Santos–Victor received the PhD degree in Electrical and Computer Engineering in 1995 from Instituto Superior Tcnico (IST - Lisbon, Portugal), in the area of Computer Vision and Robotics. He is an Associate Professor at the Department of Electrical and Computer Engineering of IST and a researcher of the Institute of Systems and Robotics (ISR), at the Computer and Robot Vision Lab - VisLab. (http://vislab.isr.ist.utl.pt) He is the scientific responsible for the participation of IST in various European and National research projects in the areas of Computer Vision and Robotics. His research interests are in the areas of Computer and Robot Vision, particularly in the relationship between visual perception and the control of action, biologically inspired vision and robotics, cognitive vision and visual controlled (land, air and underwater) mobile robots. Prof. Santos-Victor is an IEEE member and an Associated Editor of the IEEE Transactions on Robotics.