EFFICIENT COMPUTATION OF GABOR FEATURES

J. Ilonen, J.-K. Kämäräinen and H. Kälviäinen

EFFICIENT COMPUTATION OF GABOR FEATURES

J. Ilonen, J.-K. Kämäräinen and H. Kälviäinen

Efficient computation of Gabor features

J. Ilonen, J.-K. Kamarainen, H. Kälviäinen

Department of Information Technology,
Lappeenranta University of Technology,
P.O.Box 20, FIN-53851 Lappeenranta, Finland

Abstract

Gabor filter responses are widely and successfully used as general purpose features in many computer vision tasks, such as in texture segmentation, face detection and recognition, and iris recognition. In a typical feature construction the Gabor filters are utilized via multi-resolution structure, consisting of filters tuned to several different frequencies and orientations. The multi-resolution structure relates the Gabor features to wavelets, but the main difference, non-orthogonality, also is connected to the main weakness of the Gabor features: computational heaviness. The computational complexity prevents their use in many real-time or near real-time tasks, such as in object tracking. Fortunately, many arithmetic tricks exist which can be employed to significantly improve the computational complexity with negligible loss in accuracy. The main contribution of this study is the comprehensive survey of existing and development of new improvements which can be applied to filter parameter selection, filter construction, and feature computation. They are combined to provide a complete framework for optimally efficient computation of Gabor features. To make the proposed framework the most valuable and useful the implementation is distributed as public software.

1 Introduction

Gabor features constructed from post-processed Gabor filter responses have been successfully used in various important computer vision tasks, such as in texture segmentation [2], face detection [9], and iris pattern description [6]. However, only very rarely the main weakness of Gabor filter based features, the computational heaviness, has received any attention even though it may prevent the use of proposed methods in real applications. It is evident that Gabor filters have many advantageous or even superior properties for feature extraction [12], but if the computational complexity cannot be improved their application areas will remain limited.

Since Gabor filters correspond to any linear filters the most straightforward technique to perform the filtering operation is via the convolution in the spatial domain. The standard convolution with Gabor filters can be improved by utilizing the separability of Gabor filters [4, 15] or their symmetry, anti-symmetry and wavelet characteristics for reducing the number of needed multiplications and additions [17]. The convolution improvements however apply only for certain filter configurations making them merely special cases, and thus, it often occurs that the school book solution, performing filtering in the frequency domain, provides the most efficient general improvement.

In addition, certain approximation techniques yielding to more efficient computation, such as recursive Gabor approximation [18] or approximation by decomposition into Gaussians [1], have been proposed, but the approximations do not guarantee the beneficial feature space properties [12], and thus, the advantages may remain somehow artificial. Main techniques which can be used in an efficient computation, but which are hitherto neglected, are external knowledge about how the features are typically used: the multi-resolution structure utilizing several frequencies and orientations, and the stable numerical support provided by a relatively small effective area of the filters.

The main contributions of this study are (i) extensive survey of Gabor filter properties which are exploited in the literature, (ii) simple but efficient novel improvements based on the filter properties, and (iii) devising an optimal framework for computing the filter responses. An implementation of the framework is published as public software [11].

The document is divided to four main parts. In Section 2, Gabor features and simple Gabor feature space are described shortly. In Section 3, some background for efficient Gabor filtering is presented before going into details on how filter envelopes can be calculated and how some properties of Gabor filters can be used to speed up computations. Section 4 concerns implementation issues of Gabor filtering in both spatial and frequency domains, multi-resolution filtering and selection of the most efficient filtering method. Section 5 contains some experimental results on the benefits of optimizations to the computation speed and also what kind of errors the optimizations cause.

2 Constructing Gabor features

Common to all Gabor features is that they are based on Gabor filter responses for a given input image. The responses over the image are calculated for a set of filters, a bank, tuned to various orientations and frequencies. In the following the principles are briefly visited and novel methods for selecting the filter parameters are devised.

2.1 Gabor filter in 1-d

The 1-D description is included since most of the results can be conveniently generalized to 2-D filters.

The normalized Gabor filter in the time domain is [12]

$$\psi(t) = \frac{|f_0|}{\gamma\sqrt{\pi}} e^{-\frac{|f_0|}{\gamma}^2 t^2} e^{j2\pi f_o t} \tag{1}$$

where f_0 is the filter frequency and γ is the filter bandwidth. The filter bandwidth can be also considered as filter sharpness: the sharper a filter is, the narrower is the bandwidth. The Gabor filter is a complex sinusoidal wave of particular frequency modulated by a Gaussian envelope which defines the time duration. The effective time duration is inversely proportional to the effective bandwidth via the uncertainty relation.

The equation for Gabor filter in Fourier domain is [12]

$$\Psi(u) = e^{-\frac{\gamma\pi}{f_0}^2 (u-f_0)^2} \tag{2}$$

where u denotes frequency.

Examples of a Gabor filter in 1-d can be seen in Fig. 1. The filter is presented with same parameters in the both time and frequency domain. The frequency of the filter, $f_0 = \frac{1}{20} = 0.05$, is clearly seen in the frequency domain filter. In time domain the wavelength is $\frac{1}{f_0}$ units, which is 20 units in this case. Increasing value of γ would make the time domain filter wider, i.e., a larger number of waves with the wavelength 20. In the frequency domain the filter would correspondingly become sharper, i.e., more closely attuned to the filter's frequency.
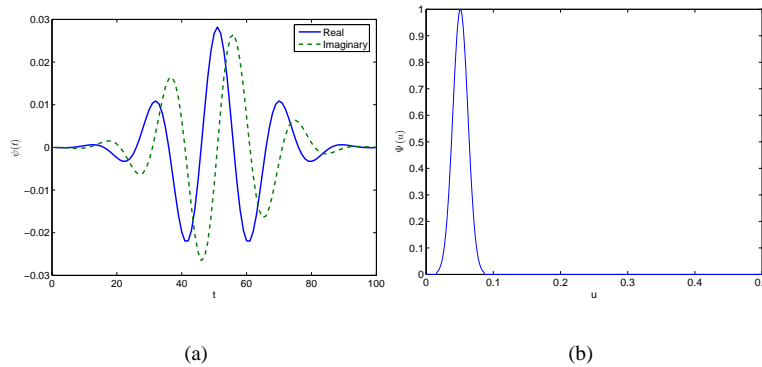

Figure 1: Gabor filter in 1-d, $f_0 = \frac{1}{20} = 0.05$, $\gamma = 1$; (a) time domain; (b) frequency domain.

2.2 Gabor filter in 2-d

2-d Gabor filter is a product of an elliptical Gaussian in any rotation and a complex exponential representing a sinusoidal plane wave. The sharpness of the filter is controlled on major and minor axis by γ and η. The filter response can be normalized to have a compact closed form [12]

$$
\begin{aligned}
\psi(x, y; f_0, \theta) &= \frac{f_0^2}{\pi \gamma \eta} e^{-\left(\frac{f_0^2}{\gamma^2} x'^2 + \frac{f_0^2}{\eta^2} y'^2\right)} e^{j 2\pi f_0 x'} \\
x' &= x \cos\theta + y \sin\theta \\
y' &= -x \sin\theta + y \cos\theta
\end{aligned}
\tag{3}
$$

where f_0 is the central frequency of the filter, θ is the rotation angle of both the Gaussian major axis and the plane wave, γ is the sharpness along the major axis and η is the sharpness along the minor axis (perpendicular to the wave). The aspect ratio of the Gaussian is $\lambda = \eta/\gamma$. The normalized Gabor filter in the frequency domain is

$$
\begin{aligned}
\Psi(u, v; f_0, \theta) &= e^{-\pi^2 \left(\frac{u'-f_0}{\alpha^2} + \frac{v'}{\beta^2}\right)} \\
u' &= u \cos\theta + v \sin\theta \\
v' &= -u \sin\theta + v \cos\theta.
\end{aligned}
\tag{4}
$$

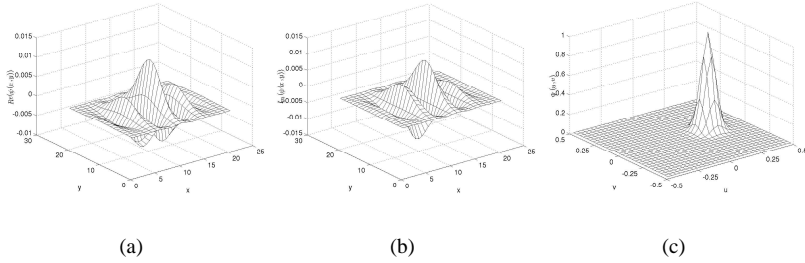Examples of Gabor filters in 2-d are presented in Fig. 2.

| (a) | (b) | (c) |

Figure 2: Gabor filter in 2-d, $f_0 = 0.2$, $\theta = 0$, $\gamma = \eta = 1$: spatial domain (a) real component; (b) imaginary component; (c) frequency domain.

The form in Eq. (4) is centered to the origin and a filter response for an image function $\xi(x, y)$ can be calculated at any location (x, y) with the convolution

$$
\begin{aligned}
r_\xi(x, y; f, \theta) &= \psi(x, y; f, \theta) * \xi(x, y) \\
&= \iint_{-\infty}^{\infty} \psi(x - x_\tau, y - y_\tau; f, \theta) \xi(x_\tau, y_\tau) dx_\tau dy_\tau .
\end{aligned}
\tag{5}
$$

2.3 Multi-resolution Gabor features

Typical Gabor feature, such as Simple Gabor feature space, consists of responses calculated with Gabor filters at several different orientations and scales (frequencies): a

filter bank. Using many different orientations and scales ensures invariance; objects can be recognized at various different orientations, scales and translations. Invariance properties of Gabor filters have been presented in more detail in [12, 13].and they will

It can be stated for an image ξ', which equals ξ rotated by ϕ, scaled by a and intensity multiplied by c, that

$$r_{\xi'}(x_0, y_0; f, \theta) = c\, r_\xi(ax_0, ay_0; \frac{f}{a}, \theta - \phi) \ , \tag{6}$$

that is, geometrical transformations of an object can be captured by filter manipulation.

A filter bank consisting of several filters needs to be used because relationships between responses provide the basis for distinguishing objects. The selection of discrete rotation angles θ_l has already been demonstrated in [16], where it was shown that the orientations must be spaced uniformly.

$$\theta_l = \frac{l2\pi}{n} \quad l = \{0, \ldots, n-1\} \ , \tag{7}$$

where θ_l is the lth orientation and n is the total number of orientations to be used. The computation can be reduced to half since responses on angles $[\pi, 2\pi[$ are complex conjugates of responses on $[0, \pi[$ in a case of a real valued input. For the result in Eq. (6) to hold the frequencies must be drawn from [12, 14],

$$f_l = k^{-l} f_{max} \quad l = \{0, \ldots, m-1\}. \tag{8}$$

Useful values for k include $k = 2$ for octave spacing and $k = \sqrt{2}$ for half-octave spacing.

Now, using the features in Eq. (5) and the parameter selection schemes to cover frequencies of interest $f_0, \ldots, f_{m-1}$ and the orientations for desired angular discrimination, one can construct a set of features at an image location (x_0, y_0). For instance in simple Gabor feature space a feature matrix $\mathbf{G}$ is used [13]

$$\mathbf{G} = \begin{pmatrix} r(x_0, y_0; f_0, \theta_0) & \cdots & r(x_0, y_0; f_0, \theta_{n-1}) \\ r(x_0, y_0; f_1, \theta_0) & \cdots & r(x_0, y_0; f_1, \theta_{n-1}) \\ \vdots & \vdots & \vdots \\ r(x_0, y_0; f_{m-1}, \theta_0) & \cdots & r(x_0, y_0; f_{m-1}, \theta_{n-1}) \end{pmatrix} \ . \tag{9}$$

The feature matrix can be used as an input feature for any classifier. If features are extracted from objects in a standard pose, it is possible to introduce matrix manipulations that allow invariant search [12, 13]. Column-wise circular shift of the feature matrix provides orientation manipulation, though if responses are calculated for only half orientation space the phase wrapping must be taken into consideration. Similarly a row-wise shift provides scale manipulation. In the case of scale manipulation the shift is not circular but the highest frequencies vanish and new lower frequencies are mapped into feature matrix.

The parameters of a multi-resolution Gabor filter bank are listed in Table 1.

2.4   Calculating filter spacing

Another application dependent problem is the selection of optimal values for filter frequencies, bandwidths and number of orientations. Optimization methods exist, but their domains are limited [12]. It is not however necessary to define all parameters presented in Table 1 separately due to their interdependencies, and since the analytical solutions have not yet been reported they will be introduced next.

Table 1: Parameters of a multi-resolution Gabor filter bank.

Parameter	Description
p_1	Crossing point between filters in adjacent frequencies
p_2	Crossing point between filters in adjacent orientations
k	Scaling factor for filter frequencies
γ	Filter sharpness along major axis
m	Number of filters in different frequencies
f_{min}	Tuning frequency of the lowest frequency filter
f_{max}	Tuning frequency of the highest frequency filter
η	Filter sharpness along minor axis
n	Number of filters in different orientations

2.4.1 Filter frequency spacing

The frequencies of filters in a filter bank are $f_0 = f_{max}$, $f_1 = f_{max}/k$, $f_2 = f_{max}/k^2$, ..., $f_n = f_{max}/k^{m-1}$. Selection of values of k and γ are interdependent: they should be selected so that the filter bank captures all frequencies important for the application, e.g., captures all frequencies to represent an object. In Figure 3, a suitable value of γ has been calculated after setting $k = \sqrt{2}$ and the crossing point between adjacent filters at $p_1 = 0.2$.
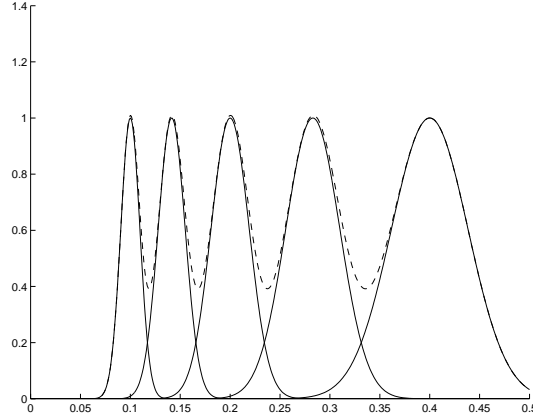

Figure 3: Sequential filters crossing at value $p_1 = 0.2$.

Using Eq. (2) a point u_a can be solved where value of the equation is p_1.

$$
\begin{aligned}
\Psi(u) & = e^{-\left(\frac{\gamma\pi}{f_0}\right)^2 (u_a - f_0)^2} = p_1 \\
& \Rightarrow u_a = f_0 \left(1 \pm \frac{1}{\gamma\pi} \sqrt{-\ln p_1} \right),
\end{aligned}
\tag{10}
$$

which corresponds to two adjacent filters at frequencies f_0 and f_0/k. Therefore,

$$f_0 \left(1 - \frac{1}{\gamma \pi} \sqrt{- \ln p_1} \right) = \frac{f_0}{k} \left(1 + \frac{1}{\gamma \pi} \sqrt{- \ln p_1} \right)$$

$$\Rightarrow \quad k = \frac{1 + \frac{1}{\gamma \pi} \sqrt{- \ln p_1}}{1 - \frac{1}{\gamma \pi} \sqrt{- \ln p_1}}. \tag{11}$$

On the other hand k and p, filter frequency scaling factor and crossing point between adjacent filters, are specified, γ can be solved from Eq. (11):

$$\gamma = \frac{1}{\pi} \left(\frac{k+1}{k-1} \right) \sqrt{- \ln p_1}. \tag{12}$$

Also p_1 can be solved from Eq.(11) when γ and k are known as

$$p_1 = e^{-\left(\gamma \pi \frac{k-1}{k+1} \right)^2}. \tag{13}$$

Additionally, we might want to solve k when $f_0 = f_{max}$, $f_{m-1} = f_{min}$ and m are given:

$$f_{min} = \frac{1}{k^{m-1}} f_{max}$$

$$\Rightarrow \quad k = e^{- \frac{\ln f_{min} - \ln f_{max}}{m-1}}. \tag{14}$$

Also an indicative value for m can be solved from Eq.(14) based on f_{max}, f_{min} and k,

$$m = - \frac{\ln f_{min} - \ln f_{max}}{\ln k} + 1. \tag{15}$$

The exact value returned by the equation is not usable directly because m is an integer.

A table which helps to calculate correct values for any combination of frequency related variables is given in Table 2.

Table 2: Parameter equations for filter frequency spacing.

p_1	k	γ	m	f_{min}	f_{max}
p_1	k	$\frac{1}{\pi} \left(\frac{k+1}{k-1} \right) \sqrt{- \ln p_1}$			
p_1	$\frac{1 + \frac{1}{\gamma \pi} \sqrt{- \ln p_1}}{1 - \frac{1}{\gamma \pi} \sqrt{- \ln p_1}}$	γ			
$e^{-\left(\gamma \pi \frac{k-1}{k+1} \right)^2}$	k	γ			
	$e^{- \frac{\ln f_{min} - \ln f_{max}}{m-1}}$		m	f_{min}	f_{max}
	k		$- \frac{\ln f_{min} - \ln f_{max}}{\ln k} + 1$	f_{min}	f_{max}
	k		m	$\frac{1}{k^{m-1}} f_{max}$	f_{max}
	k		m	f_{min}	$f_{min} k^{m-1}$

2.4.2 Filter orientation spacing

The minor axis sharpness of a 2-d Gabor filter, η, can be calculated based on the number of orientations and required overlap. In Fig. 4 a diagram of two Gabor filters in the frequency space is shown.
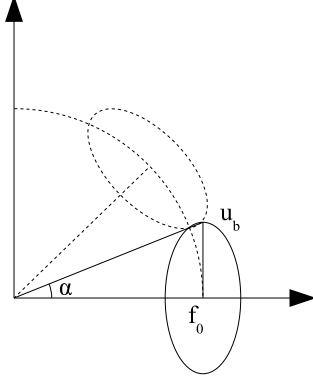
Figure 4: Two Gabor filters with different orientations in the frequency space.

Solving η is based on Eq. (2) with a crossing point p_2 between two filters in adjacent orientations:

$$\begin{aligned} \Psi(u) &= e^{-\left(\frac{\eta\pi}{f_0}\right)^2 u_b^2} = p_2 \\ &\rightarrow \quad \eta = \frac{f_0}{\pi}\frac{\sqrt{-\ln p_2}}{u_b} \end{aligned} \tag{16}$$

Now, u_b can be solved from $u_b = \tan\left(\frac{\pi}{2n}\right) f_0$, where n is the number of filter orientations. However, this creates needlessly wide filters when number of filter orientations is small, $n < 4$. Another possibility is to use an approximation for u_b by dividing circumference of a circle by number of filters, $u_b = \frac{\pi f_0}{2n}$. Therefore, η can be solved to either

$$\eta = \frac{1}{\pi}\frac{\sqrt{-\ln p_2}}{\tan\left(\frac{\pi}{2n}\right)} \quad \text{or} \quad \eta = \frac{1}{\pi}\frac{\sqrt{-\ln p_2}}{\frac{\pi}{2n}}. \tag{17}$$

When number of orientations, n, is large, η calculated by both of the equations approaches the same value, but with a small n, the first solution for u_b leads to needlessly wide filters, so the latter equation is preferred. With approximate u_b p_2 can be solved from Eq. (16) as

$$p_2 = e^{-\left(\frac{\eta\pi^2}{2n}\right)^2}. \tag{18}$$

Additionally an indicative value for n can be solved based on p and η,

$$n = \sqrt{-\frac{\left(\eta\pi^2\right)^2}{4\ln p_2}}. \tag{19}$$

Actual value must be an integer.

A table which helps to calculate correct values for any combination of orientation related variables is given in Table 3.

8

Table 3: Parameter equations for filter orientation spacing.

p_2	η	n
p_2	η	$\sqrt{-\dfrac{(\eta\pi^2)^2}{4\ln p_2}}$
p_2	$\dfrac{1}{\pi}\dfrac{\sqrt{-\ln p_2}}{\frac{\pi}{2n}}$	n
$e^{-\left(\frac{\eta\pi^2}{2n}\right)^2}$	η	n

2.4.3 Example of filter spacing

Two filter banks in the frequency space are presented in Fig. 5. Only the upper half of the filter bank is needed because responses on the lower half are complex conjugates. In Fig. 5(a) filters are closely located in the frequency space ($k = \sqrt{2}$) and therefore in the frequency direction the filters are sharp (γ is large). The same value was used for η and consequently there are large gaps between filters in different orientations, and as a result there can be features at specific angles that cannot be detected by the filter bank in Fig. 5(a). In Fig. 5(b) η is solved and the gaps between filters in different orientations disappear.
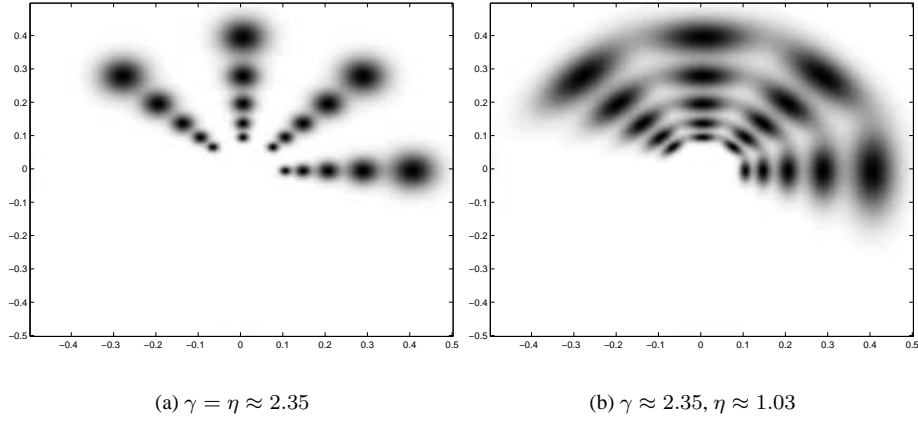

(a) $\gamma = \eta \approx 2.35$ (b) $\gamma \approx 2.35, \eta \approx 1.03$

Figure 5: Examples of filter banks in frequency space, both use $m = 5$, $n = 4$, $p = 0.2$ and $k = \sqrt{2}$; (a) $\gamma = \eta \approx 2.35$; (b) $\gamma \approx 2.35, \eta \approx 1.03$.

3 Efficient Gabor Filtering

In this section the most important characteristics of Gabor filters and filtering in the both domains are discussed in the context of computational complexity. It should be noted that the results are mainly devised in the discrete domain and in order them to apply the discrete Gabor filter construction must not violate restrictions given in [12].

3.1 Background

3.1.1 Convolution

Convolution of two functions $f(x)$ and $g(x)$ is

$$h(x) = f(x) * g(x) = \int_{-\infty}^{\infty} f(u)g(x-u)du \ . \tag{20}$$

Denoting that $F(s)$ is the Fourier transform of $f(x)$ and $G(s)$ is the Fourier transform of $g(x)$, then $f(x) * g(x)$ has the Fourier transform $F(s)G(s)$. This result is known as the convolution theorem. The convolution can then be calculated as (a bar denotes Fourier transform, and a long bar denotes inverse Fourier transform):

$$f * g = \overline{\overline{f}\overline{g}} = f * g, \tag{21}$$

i.e., the convolution of two functions is the inverse transform of the product of their transforms [3].

In practice discrete signals are used and for a discrete signal g and convolution mask f of length M Eq. (20) becomes

$$h(m) = \sum_{u=-M/2}^{M/2} f(n)g(m-n). \tag{22}$$

For calculating convolution in one point of the signal $O(M)$ calculations are needed. If the length of signal f is N, the total complexity is $O(MN)$. By using Eq. (21) the convolution of whole signal is achieved by taking Fourier transforms of both the signal and the convolution mask, multiplying them and taking inverse Fourier transform. FFT has complexity of $O(N \log N)$, and thus, the complexity of convolution becomes approximately $O(3N \log N)$. Unless M is very small (a small mask), the latter method is faster. Convolution can be straightforwardly extended to 2-dimensional signals in which case the complexities become (with a square signal and mask) $O(M^2 N^2)$ and $O(3N^2 log N)$ for simple convolution and the Fourier transform based method, respectively.

3.1.2 Fourier transform

The discrete Fourier transform (DFT) is the foundation for the algorithms working in frequency domain, including convolution. The Fourier transform, $F(u)$, for a discrete function $f(x)$ is

$$F(u) = \frac{1}{N} \sum_{x=0}^{N-1} f(x)e^{-j2\pi ux/N}, \text{for } u = 0, 1, 2, ..., N-1 \tag{23}$$

where $j = \sqrt{-1}$. Conversely the inverse transform is

$$f(x) = \sum_{u=0}^{N-1} F(u)e^{j2\pi ux/N}, \text{for } u = 0, 1, 2, ..., N-1. \tag{24}$$

The complexity of DFT is $O(N^2)$: for every value of $F(u), u = 0, 1, ..., N-1$ a summation for all values of $f(x), x = 0, 1, ..., N-1$ is needed [8].

A more efficient algorithm for calculating DFT exists: FFT (Fast Fourier transform). FFT has sequential complexity of $O(N \log N)$. The DFT equation can be written as

$$F(u) = \frac{1}{N} \sum_{x=0}^{N-1} f(x)w^{xu} \tag{25}$$

where $w = e^{-j2\pi/N}$ is called as a twiddle factor. There exists many formulations for FFT, but generally the algorithm works by dividing the summation recursively into smaller parts by a divide-and-conquer approach. The summation can be divided to two parts

$$F(u) = \frac{1}{N} \left[\sum_{x=0}^{N/2-1} f(2x)w^{2xu} + \sum_{x=0}^{N/2-1} f(2x+1)w^{(2x+1)u} \right] \tag{26}$$

where the first part corresponds to f with even indexes and the second part to odd indexes. The equation can be rearranged as

$$F(u) = \frac{1}{2} \left[\frac{1}{N/2} \sum_{x=0}^{N/2-1} f(2x)w^{2xu} + w^u \frac{1}{N/2} \sum_{x=0}^{N/2-1} f(2x+1)w^{2xu} \right]. \tag{27}$$

Each summation is now $N/2$ point DFT and the equation can be written as

$$F(u) = \frac{1}{2} \left[F_{\text{even}} + w^u F_{\text{odd}} \right]. \tag{28}$$

Each $N/2$ point DFT can be divided to two $N/4$ point DFTs and the decomposition can be continued until single points are to be transformed, leading to overall complexity $O(N \log N)$. Transforming a 2-d signal is done in two steps: at first, the FFT is performed for all image rows, and then, the FFT is performed for all columns of the row-wise transformed image. It does not matter whether row- or column-wise FFT is performed first. For an image of size $N \times N$ $2N$ FFT transforms with complexity $O(N \log N)$ need to computed, and therefore, the total complexity is $O(2N^2 \log N)$.

The generic FFT formulation is limited to working with signals of length $2^n, n = 1, 2, ...$ but fast solutions for other signal lengths has been devised. One example of a highly optimized FFT library is FFTW3 [7], which implements the best known FFT formulations and utilizes a sophisticated code-generation framework for finding the most effective transformation method for different hardware platforms. The library can compute FFT in $O(N \log N)$ time for a signal of any length, and the code generation framework enables automatic optimization for hardware platforms with varying memory bandwidth, cache, and floating point computation performance characteristics.

3.2 Effective filter envelopes

Effective filter envelope corresponds to a support area where filter coefficients are significant. Coefficients outside this area can be discarded with only negligible effect in accuracy. Since Gabor filters are elliptical functions the effective envelope is an ellipse which can be encapsulated by a minimal size rectangle. The size of the rectangle may significantly reduce the computational complexity in the spatial domain filtering and save memory in the frequency domain filtering.

The support of a Gabor filter is infinite, but in the discrete domain the filter size is always limited. Requirements for accurate filtering have been given in [12], but here the requirements are revised and further applied to reduce the computational complexity.

Effective filter envelope is defined by the Gaussian part of the Gabor filter, the sinusoidal part can be ignored. The envelope for a 1-d filter is the shortest interval of the filter that includes some defined percent of the total filter energy. For a 2-d filter the envelope is similarly the smallest area (an ellipse) which includes certain percent of the total filter energy [10].

3.2.1 Envelopes in 1-d

The effective filter envelope – the shortest interval of the filter that contains a defined part of the total filter energy – contains the most important part of the filter, therefore it is unnecessary to calculate filter in other areas. Time domain filtering is based on the convolution which has complexity $O(M)$ for a single point (here, a single Gabor filter response), where M is the size of the filter. For a smaller filter the responses can be computed faster. Gabor filter, Eq. (1), consists of two parts: a sinusoidal wave and a Gaussian envelope. For solving the effective filter envelope, only the Gaussian part of the filter has to be taken into account. The Gaussian equation cannot be integrated analytically, but the effective filter envelope must be solved using approximation methods.

The envelope has the standard Gaussian form,

$$g(t) = \frac{1}{\sigma\sqrt{2\pi}} e^{-\frac{(t-\mu)^2}{2\sigma^2}}. \tag{29}$$

which area cannot be analytically integrated but effective numerical methods exist, and therefore, the form can be used to estimate envelopes. The corresponding substitutions can be solved for the spatial domain filter, Eq. (1), as $\mu = 0$ and

$$\sigma = \frac{\gamma}{\sqrt{2}|f_0|}. \tag{30}$$

For the frequency domain filter, Eq. (2), the substitutions are $\mu = f_0$ and

$$\sigma = \frac{f_0}{\gamma\pi\sqrt{2}}. \tag{31}$$

3.2.2 Envelopes in 2-d

In 2-d the effective interval is replaced by an effective ellipse and consequently by a minimal rectangle where the ellipse fits. In the integration the separability of Gaussian can be utilized, i.e, it is sufficient to define intervals including e_{1d} percent of the filter energy in two perpendicular directions and multiply them. The envelope will be a lower

bound estimate for the envelope, i.e. it will contain more energy than an accurately solved ellipsoidal envelope.

There are two 1-d Gaussians directed along the filter's major and minor axes. The Gaussians are controlled by two different sharpness values, γ and η, and for the two Gaussians 1-d filter envelopes are solved separately, i.e, the interval for including e_{1d} percent of the filter energy. Calculation of 1-d envelopes was explained in Section 3.2.1. When two 1-d envelopes are combined to create a 2-d envelope, the contained energy will be $e_{2d} = e_{1d}^2$. Two 1-d envelopes are used to create an approximate square envelope. Additionally, the square envelope must be rotated by θ and a new, final envelope is created.

The 1-d envelopes form the major and minor axes of the ellipsoidal envelope, a and b. Now, a rectangular envelope can determined by finding the points in the ellipse which are directed along x and y axes after the rotation (see Fig. 6). These points are the points in the derivative of the ellipse's equation with slopes $\tan\theta$ and $-\tan(\frac{\pi}{2}-\theta)$. The equation for ellipse with major axis a and minor axis b is

$$\frac{a^2}{x^2} + \frac{y^2}{b^2} = 1, a > 0, b > 0 \Rightarrow y = \pm\frac{b}{a}\sqrt{a^2 - x^2}. \tag{32}$$

After derivation the equation becomes

$$y' = \pm\frac{-bx}{a\sqrt{a^2 - x^2}}. \tag{33}$$

Now, point with specific slope must be found, $y' = c$,
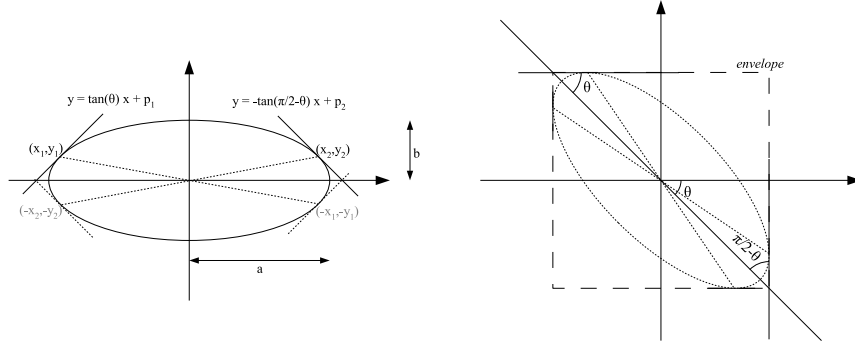
$$x = \pm\frac{ca^2}{b^2 + c^2 a^2}. \tag{34}$$

Figure 6: Determining effective envelope for a 2-d Gabor filter in spatial domain. Major and minor axes, a and b, of the ellipsoidal envelope must be known as well as filter's orientation θ.

Four points, (x_1, y_1), $(-x_1, -y_1)$, (x_2, y_2), and $(-x_2, -y_2)$, lie in the border of the envelope. To get the final envelope the points must be rotated in relation to the origin by θ,

$$A_s = \begin{bmatrix} x_1 & y_1 \\ -x_1 & -y_1 \\ x_2 & y_2 \\ -x_2 & -y_2 \end{bmatrix} \begin{bmatrix} \cos\theta & \sin\theta \\ -\sin\theta & \cos\theta \end{bmatrix}. \tag{35}$$

Now, smallest and largest x and y coordinates must be selected from A_s. These coordinates define the rectangular envelope. If $\theta = n\frac{\pi}{2}, n = 0, 1, 2, ...$ one of the slopes goes to infinity, and the four points for defining the envelope before rotation are $(a, 0)$, $(-a, 0)$, $(0, b)$, and $(0, -b)$. In the frequency domain the envelope is calculated similarly, except that the 1-d envelopes leading to a and b must be solved for frequency domain case (see Section 3.2.1) and the filter is not centered at the origin but the location is defined by the filter's frequency f_0, and the the ellipse is centered in $(f_0, 0)$,

$$A_f = \begin{bmatrix} f_0 + x_1 & y_1 \\ f_0 - x_1 & -y_1 \\ f_0 + x_2 & y_2 \\ f_0 - x_2 & -y_2 \end{bmatrix} \begin{bmatrix} \cos\theta & \sin\theta \\ -\sin\theta & \cos\theta \end{bmatrix}. \tag{36}$$

The actual envelope is again determined by the smallest and largest x and y coordinates in A_f.

3.2.3 Determining the highest frequency of the filter

The frequency domain envelope has an important property: the higher bound defines also the highest needed frequency, f_{high}. This information can be used to downscale the signal prior to filtering. Depending on how low the highest frequency is the downscaling may have a very significant effect to the computation speed: the image to be filtered is now smaller and can be filtered faster. Also the size of a spatial domain filter is reduced. An input image can be downscaled before filtering by a scaling factor

$$a_{sf} = \frac{0.5}{f_{high}}, \tag{37}$$

where 0.5 is the Nyquist frequency. As a result, the unnecessary frequencies residing above f_{high} are removed and computations needed for filtering are reduced, since both the size of the image and the convolution mask are now smaller. It must be noted that downscaling might not be always wanted depending on the final use of the filter responses. For instance, in case of object detection the accuracy may suffer with low resolution responses if the object to be detected lies in the border of two adjacent pixels in the low resolution image.

To solve f_{high} the maximum distance from the origin to the edge of the ellipsoid envelope of the Gabor filter in the frequency domain must be found. The center of the ellipse is located in point $(f_0, 0)$, where f_0 is the frequency of the filter, and its major axis is a and minor b. The distance from the origin to the edge of the ellipse is

$$d(x) = \sqrt{(f_0 + x)^2 + \left(\frac{b}{a}\sqrt{a^2 - x^2}\right)^2}, |x| \le a. \tag{38}$$

The concept is illustrated in Fig. 7. The lower half of the ellipse can be ignored since it is symmetrical to the upper half.

If $b < a$, $f_{high} = f_0 + a$ because no point in the edge of the ellipse can be farther from the origin than the edge-point of the envelope residing in the x-axis. For case
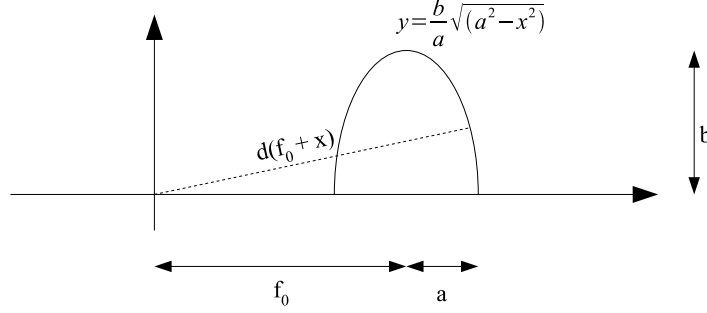
14

Figure 7: Determining f_{high} with the help of an ellipsoidal envelope of a 2-d Gabor filter in the frequency domain.

$b > a$ the maximum of $d(x)$ must be solved by finding the zero-point of the derivative of $d(x)$,

$$d'(x) = \frac{2f_0 + 2x - \frac{2b^2 x}{a^2}}{2\sqrt{(f_0 + x)^2 + \left(\frac{b}{a}\sqrt{a^2 - x^2}\right)^2}} = 0, \tag{39}$$

which leads to

$$x = -\frac{a^2 f_0}{a^2 - b^2}, b > a, |x| \le a. \tag{40}$$

The equation may give solution $x > a$ in which case $f_{high} = f_0 + a$, otherwise f_{high} can be found by applying x in Eq. (38), $f_{high} = d(x)$.

3.3 Separability of Gabor filters

If a filter G can be expressed as multiplication of two vectors, $G_{col} * G_{row}$, the filter G is separable. For separable filters the convolution can be performed separately with 1-dimensional filters G_{col} and G_{row}. This decreases the total complexity from $O(M^2 N^2)$ to $O(2MN^2)$, where N is the width/height for the image and M for the filter. As compared to FFT filtering, $O(N^2 \log N)$, it can be noted that the spatial domain convolution with separable filters is beneficial when $M < \log N$.

Gabor filters which are parallel (horizontal and vertical) to the image axes are separable, $\theta = n\frac{\pi}{2}, n = 0, 1, 2, \dots$. One filter is a sinusoidal function with Gaussian envelope and the another a Gaussian envelope. If filters with arbitrary orientations are used, exploiting separability requires image rotation which increases complexity especially because interpolation have to be used to avoid aliasing effects. However, separable Gabor filters can extended to work with filters with $45°$ degree angle, $\theta = \frac{\pi}{4} + n\frac{\pi}{2}, n = 0, 1, 2, \dots$, which has been done in [15]. The separability is achieved by going through the image in diagonals instead of along image axes. the

3.4 Exploiting filter symmetry

Symmetry and anti-symmetry features of 2-d Gabor filters were used to speed up the spatial domain computation in [17]. Gabor filters are symmetric: the same filter values

15

will be repeated in several locations. For example in Fig. 8 is a Gabor filter where value $a = 0.0620$ or its negation is repeated four times around the center of the filter. Using generic convolution the computation concerning those four points would be $av_1 + av_2 + (-a)v_3 + (-a)v_4$ which includes four multiplications and three additions. The same can be calculated as $a(v_1 + v_2 - v_3 - v_4)$, which reduces the number of multiplications to one. Similar re-ordering can be performed for many of the filter locations leading to considerable improvement and it can be applied separately to both real and imaginary part of the filters. The symmetry and anti-symmetry properties can be used automatically to reduce multiplications when computing filter responses. However, with today's processors multiplications are not expensive operations and therefore these optimizations are not very significant anymore.
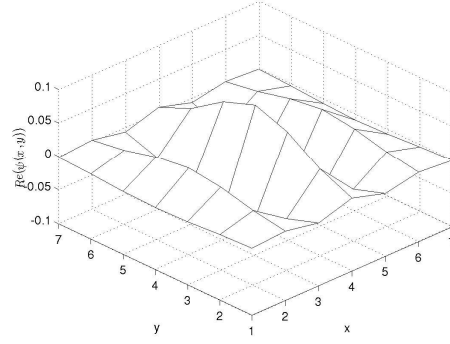
(a)

-0.0009	0.0031	-0.0065	0.0084	-0.0065	0.0031	-0.0009
-0.0031	0.0108	-0.0228	0.0293	-0.0228	0.0108	-0.0031
-0.0065	0.0228	-0.0483	0.0620	-0.0483	0.0228	-0.0065
-0.0084	0.0293	-0.0620	0.0796	-0.0620	0.0293	-0.0084
-0.0065	0.0228	-0.0483	0.0620	-0.0483	0.0228	-0.0065
-0.0031	0.0108	-0.0228	0.0293	-0.0228	0.0108	-0.0031
-0.0009	0.0031	-0.0065	0.0084	-0.0065	0.0031	-0.0009

(b)

Figure 8: (a) Real part of a Gabor filter in spatial domain; (b) real values of the Gabor filter.

4 Implementation

This section concentrates on efficient implementation details of multi-resolution Gabor filtering in both spatial and frequency domains.

4.1 Filtering in spatial domain

A diagram of filtering in spatial domain is presented in Fig 9. The complexity of convolution depends directly on the size of the convolution mask which in this case is the Gabor filter. Complexity for calculating filter response for one point is $O(M^2)$ where M is the width and height of the mask. If the filtering is done for the whole image, the complexity is $O(M^2N^2)$ where N denotes both width and height of the image. It is important for fast computation that the size of the filter, M, is as small as possible (Section 3.2.2).
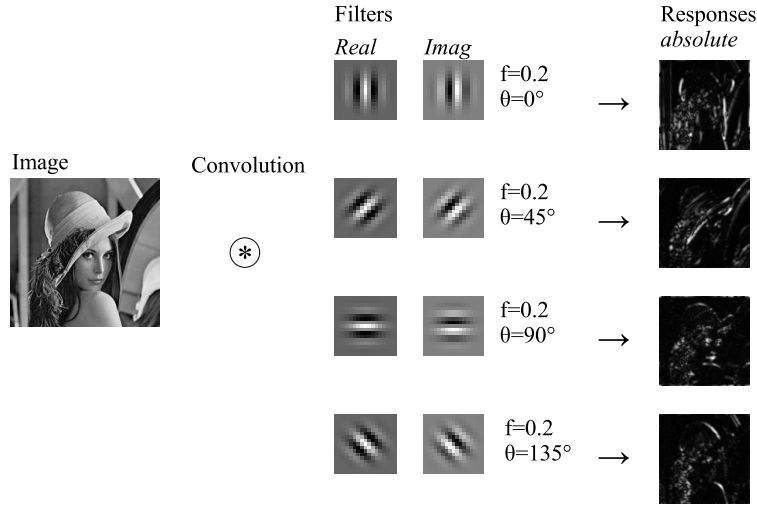
Figure 9: Filtering in spatial domain.

If the highest frequency of the filter allows, the image can be downscaled before filtering by the factor a_{sf} from Eq. (37). In practice, it is most straightforward to use downscaling only to integer factors or even only to power of two factors since then downscaling corresponds to an average of a group of pixels and no interpolation is needed to avoid aliasing effects. The complexity of downscaling by averaging is $O(N^2)$: every pixel of the original $N \times N$ image is processed to produce N^2/a_{sf}^2 pixels to the resulting image. The complexity of computing a single filter response is now $O(M^2/a_{sf}^2)$. The complexity of filtering all pixels in the downscaled image is

$$O(\frac{N^2M^2}{a_{sf}^4}). \tag{41}$$

Clearly, the complexity is drastically affected by the scaling factor a_{sf} which in turn is defined by the highest frequency of the filter f_{high}. The total complexity of computing K filter responses for a Gabor filter and by utilizing downscaling is

$$O(N^2 + K\frac{M^2}{a_{sf}^2}).$$ (42)

There are two separate steps in filtering: creation of a filter and filtering an image with the filter. An algorithm for spatial domain filter creation is presented in Alg. 1, and an algorithm for filtering in the spatial domain in Alg. 2. Scaling factor a_{sf} can be set manually to a value lower than the one determined with f_{high}. The scaling factor must be decided before the filter creation.

Algorithm 1 *Create a spatial Gabor filter with parameters f, θ, γ and η.*

1: *Solve f_{high} using f, γ and η (Section 3.2.3).*
2: *Adjust f by a_{sf}, $f' = a_{sf}f$, Eq. (37).*
3: *Solve filter envelope E for a filter with parameters (f', θ, γ, η) (Section 3.2.2).*
4: *Compute the filter g for filter area E with parameters (f', θ, γ, η).*

Algorithm 2 *Filter an image s in the spatial domain with a filter g (scaling factor a_{sf}) at locations $P = \{(x, y)_k\}$.*

1: *Downscale the image s by factor a_{sf}, $s \to s'$.*
2: **for** *All points p in P* **do**
3:     *Adjust the point's coordinate, $p' = p/a_{sf}$.*
4:     *Compute response $r(p)$ by convolving the image s' in the point p' with the filter g.*
5: **end for**

If the symmetry and anti-symmetry properties are to be used a more complex data-structure to hold the filter is needed instead of a simple 2-dimensional filter g. In practice, the complex structure would hinder the efficiency of the filtering, and therefore, the filtering algorithm needs to include a hard-coded filter for each different set of filter parameters to gain any improvement from the symmetry and anti-symmetry properties. This is not very practical for a generic filtering framework, but can be applied for a specific filtering task.

Utilizing separability would require only a small change to the filter creation algorithm Alg. 1. The filter creation would be changed so that if the filter orientation allows, i.e. $\theta = n\frac{\pi}{2}, n = 0, 1, 2, ...$, two 1-dimensional filters, g_1 and g_2, would be created instead of one 2-dimensional filter g. However, the filtering algorithm Alg. 2 would require more thorough changes, because separable filters do not decrease complexity at all for filtering at a single point. The whole image needs to be filtered, first to one direction with filter g_1 and then to perpendicular direction with filter g_2.

4.2 Filtering in frequency domain

A diagram of filtering in the frequency domain is presented in Fig 10. First, the image is converted to the frequency domain with FFT, the FFT transformed image is multiplied by a Gabor filter and the responses are converted back to the spatial domain with the inverse FFT. The complexity of 2-d FFT and IFFT is $O(N^2 \log N)$ with a constant multiplier. The multiplier depends on the image dimensions and the implementation; dimensions with low-prime factors are the fastest to compute. Slow implementations may use generic DFT (discrete Fourier transform) with complexity of $O(N^2)$ if N is

not a power of two. In this study FFTW3 library, as a part of Matlab, has been used. The library computes FFT for a signal of any length in $O(N^2 \log N)$ [7].
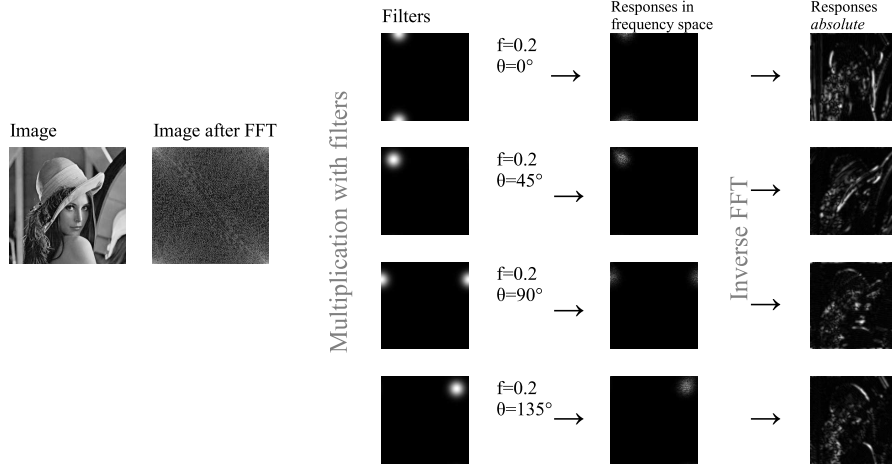

Figure 10: Filtering in frequency domain.

In this case the smallest possible size of the effective filter envelope is not as crucial as in the spatial domain case since the whole image must be converted to the frequency domain and back whether the filter envelope is used or not. However, most of the filter values will be close to zero and they can be omitted to minimize memory usage and the number of multiplications.

The image to be filtered can be downscaled before filtering similarly as in the spatial domain filtering case. Another possibility is to perform downscaling during IFFT phase of filtering. While downscaling has lower complexity than IFFT, the latter may be preferable in practice with multi-resolution filtering because the image has to be converted only once to the frequency domain. If regular downscaling is performed, FFT has to be performed for every downscaled image.

A diagram of downscaling in frequency space is presented in Fig. 11. The parts of the transformed image corresponding to higher frequencies than f_{high} can be simply discarded from the frequency domain presentation. The filter actually includes only a small portion of the frequency space even after discarding the high frequencies, however, the frequency space cannot be dissected arbitrarily and some $-f_d \rightarrow f_d$ interval must be included. In the spatial domain filter responses will have the same energy (the sum of response magnitudes will not change) when downscaling is performed in this fashion. To scale the filter response magnitudes to the correct scale they must be multiplied by $\frac{1}{a_{sf}^2}$ where a_{sf} is the scaling factor (Eq. (37)).

By downscaling with scaling factor a_{sf} the complexity of the IFFT can be reduced to

$$O(\frac{N^2}{a_{sf}^2} \log \frac{N^2}{a_{sf}^2}). \tag{43}$$

Algorithm for creating a filter in the frequency domain is presented in Alg. 3 and an algorithm for filtering in Alg. 4. Scaling factor a_{sf} can be set to any value during
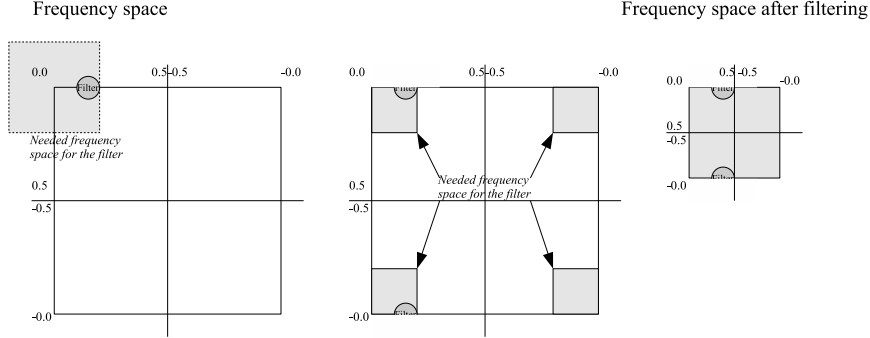
19

Frequency space   Frequency space after filtering

Figure 11: Downscaling during filtering in frequency space. Note that zero frequency is in the upper left-hand corner of the figures.

filtering with no effect to the filter creation.

Algorithm 3 *Create a frequency domain Gabor filter with parameters f, θ, γ and η.*

1: *Solve filter envelope E with parameters (f, θ, γ, η) (Section 3.2.2).*
2: *Compute the filter g for filter area E with parameters (f, θ, γ, η).*
3: *Solve f_{high} using f, γ and η (Section 3.2.3).*
4: *Solve scaling factor a_{sf}, Eq. (37).*

Algorithm 4 *Filter an image s in the frequency domain with a filter g (scaling factor a_{sf} and filter area E).*

1: *Initialize r' to the same size as s and zero.*
2: *Compute FFT of the image, $s' = F(s)$.*
3: *Filter in filter area, $r'(E) = s'(E) * g$.*
4: *Crop frequencies above $\frac{0.5}{a_{sf}}$ out of r'.*
5: *Transform responses back to spatial domain with IFFT, $r = F^{-1}(r')$.*
6: *Scale response magnitudes, $r = r\frac{1}{a_{sf}^2}$.*

4.3 Multi-resolution filtering

Multi-resolution feature extraction here is based on similar structure to Laplacian pyramid [5]. A Laplacian pyramid represents an image as a pyramid of quasi-bandpassed images (see Fig. 12), where the bottom of the pyramid has the highest frequency content of the image sampled densely, and the higher levels contain lower frequency information sampled at sparser densities. Each level of the pyramid reduces the filter band limit by an octave, and the sample density can be reduced by the same factor. The Laplacian pyramid was used for image compression, but multi-resolution Gabor feature extraction, such as simple Gabor feature space [13], uses a similar idea. Both computation time and memory will be saved as the responses are computed at lower resolutions than the original image resolution.

Implementing similar multi-resolution structure with Gabor filtering is straightforward. Alg. 4 can be used as an example: when the scaling factor a_{sf} is selected based
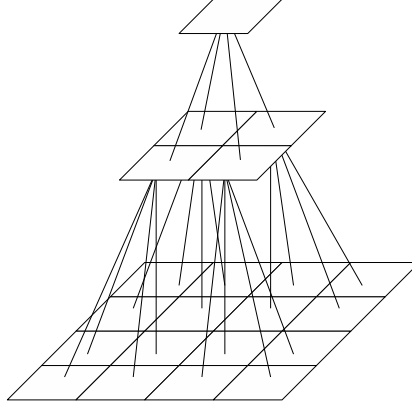
Figure 12: The structure of Laplacian pyramid.

on f_{high} of the current filter, resolution of the responses depends directly on the frequency of the filter. High frequency filter responses will be sampled densely and lower frequencies increasingly more sparsely, i.e, at lower resolution. Octave spaced filter frequencies, $k = 2$, must be used if a similar structure as in Fig. 12 is wanted: four high frequency responses correspond to one response in the next level. If some other value for k is used, the pyramid structure will not have as clear correspondences between responses in different levels.

Using a multi-resolution structure may be problematic for the following processing steps after feature extraction, for example, classification in object recognition. If the responses from filters in different frequencies must be eventually used with the same resolution, sparsely sampled responses from low frequency filters must be upscaled back to a higher resolution. Upscaling can create various errors, which are discussed in Section 5.2.2. If the the responses must be in the same resolution, it is preferable to compute all responses directly at the same resolution and omit the upscaling procedure. The resolution can be selected based on the highest frequency filter.

With multi-resolution filtering filters can be re-used in the spatial domain. Only the highest frequency filter is needed. Lower frequency responses can be computed using the same filter for downscaled image, and the process can be continued to lowest frequencies (see Fig. 13). Re-using the filters is not important for standard workstations. Filters just consume a small amount of memory and the computing time is not reduced. However, re-using filters can be important in a hardware-based solution.

4.4   Selecting the optimal filtering approach

The decision whether the filtering should be performed in the spatial or the frequency domain should be based mainly on for how many points the filter responses are required. If responses are needed for all points, frequency domain filtering is practically always preferable as can be seen comparing Eqs. (41) and (43): the $\log$-clause of the latter is very likely to be smaller than M^2/a_{sf}^2 in the first one.

Filtering all points is not always necessary, and it is worthwhile to create a decision criteria for reaching the maximum efficiency. A decision tree for selecting a proper filtering technique is presented in Fig. 14. After initial selection on whether the fre-
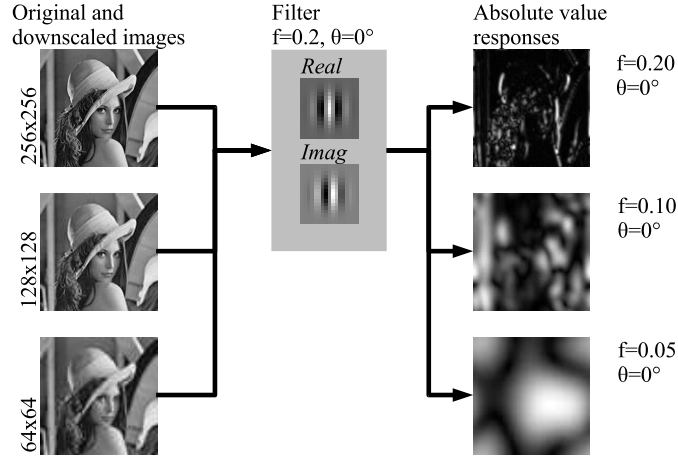
21

Figure 13: Re-using filters in the spatial domain.

quency or the spatial domain filtering is preferable, the frequency domain filtering is straightforward, but for the spatial domain the benefit of downscaling must be considered. Actual computing times may vary from the given complexities depending on the implementation details and in the case of FFT on the actual exact algorithmic complexities which depend on the signal dimensions. Still, the implementation details and given complexities should cause only a linear or constant difference to the decision tree, and the tree can be adjusted based on experiments.
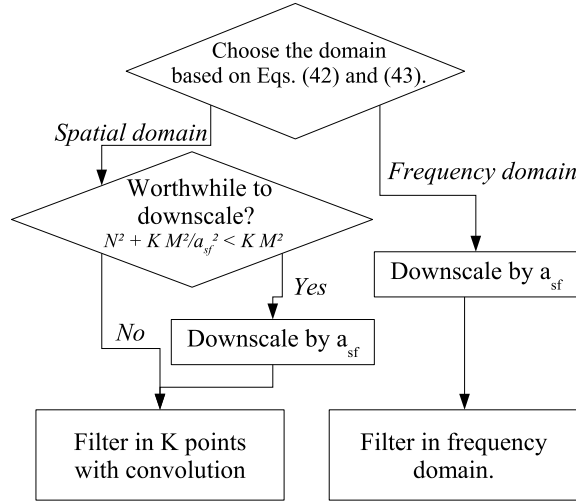

Figure 14: A decision tree for selecting an optimal filtering procedure.

5 Experiments

The authors have created a Matlab toolbox for efficient computation of Gabor filter responses [11]. The toolbox especially supports multi-resolution feature structures which are the most popularly used. Some experimental results of potential speedups and selection of filtering method are presented here.

5.1 Selecting the optimal filtering domain

The selection between the domains is demonstrated in Fig. 15. The decision tree presented in Fig. 14 would, in this case, use a cut-off point where the computation is changed from the spatial to the frequency domain at 13000 for the small (11×11) filter and to 1800 for the large (29×29) filter. In practice, for the small filter size the change should be done much earlier (after 3000 points), but for the large filter size the number is accurate. There is a rather large additional cost included to the computation of each filter response, which can be explained by the inefficiency of the implementation in Matlab environment. With a small filter the filtering is fast, and therefore, the inefficiency of the environment causes comparably larger slowdown than with a large filter where filtering itself is taking a longer time.
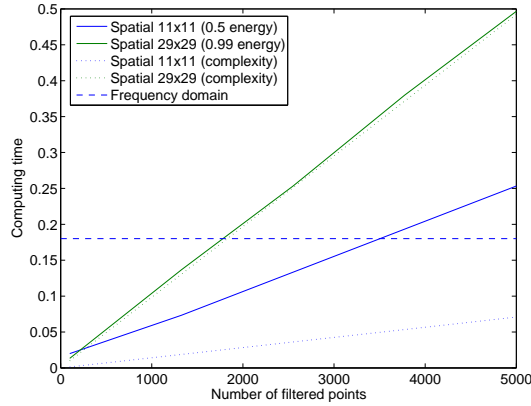

Figure 15: Computing times and complexities for calculating responses for a certain number of points.

5.2 Errors caused by optimizations

5.2.1 Effective envelopes

Using effective filter envelopes causes some errors to the filter responses as shown in Fig. 16. First, a filter bank (four orientations and four frequencies between 0.2 and 0.0707) was created with full filters, and the responses were compared to a filter bank created utilizing effective envelopes. The test image contained Gaussian noise. In both the spatial and the frequency domains the error gets steadily smaller when the filter energy (envelope size) increases. Another expected fact is that filtering time increases heavily in the spatial domain when large envelopes are used, while the envelope size has only negligible effect in the frequency domain.
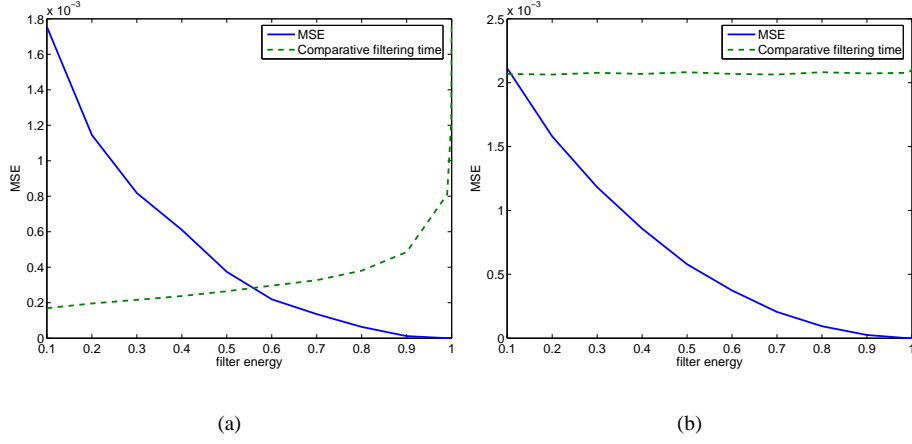
Figure 16: MSE between filter responses calculated at full energy and using an effective envelope, and the effect to filtering time; (a) spatial domain; (b) frequency domain.

With frequency domain filtering large envelopes should always be used since large filters do not significantly increase filtering time. Full size filters, however, waste enormous amount of memory, and therefore, the envelope should be set to as high as $0.99 - 0.999$. In the spatial domain the selection of suitable energy for the filter envelope is not as easy since a compromise between the accuracy and computing time must be done. However, filter energy of $0.8 - 0.9$ seems to be a point where accuracy is reasonable without computing time yet growing excessively.

5.2.2 Multi-resolution

Filter responses of the same filter with different scaling-levels are presented in Fig. 17 and 18. The downscaled filter responses look similar to responses at original resolution, as they should.
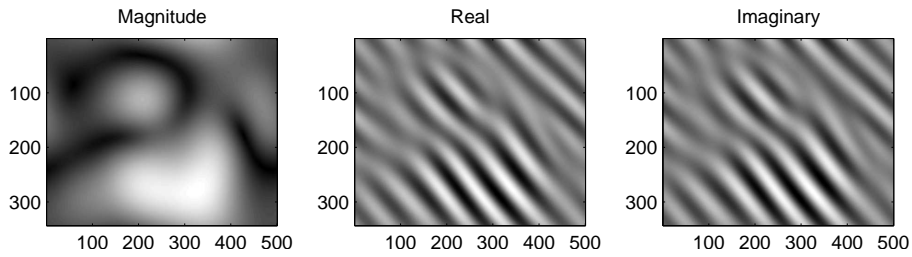

Figure 17: Filter responses from a filter at $f = 0.0177$ and orientation $135°$.

However, when the downscaled filter responses are upscaled back to the original size anomalies appear. When the downscaled filter responses, real and imaginary parts, are upscaled separately with bilinear filtering, the resulting absolute value image has a very noticeable grid pattern (Fig. 19(a)). The grid pattern occurs because the low-resolution filter responses are alternating between high and low values in neighboring
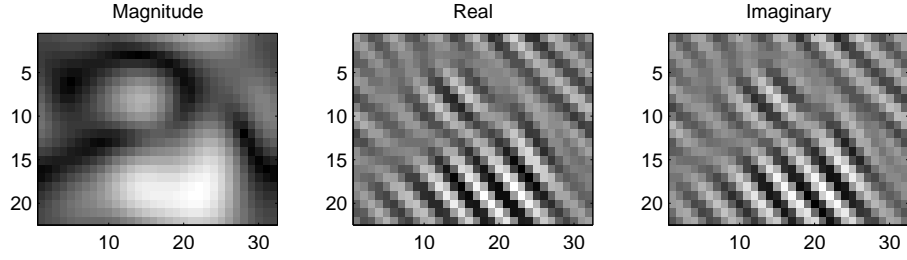
24

Figure 18: Filter responses from a filter at $f = 0.0177$ and orientation $135°$ with image downscaled to 1/16th.

pixels with no intermediate values in-between. Additionally, real and imaginary values are in different phase. The effect is visualized in Fig. 20 where only 100th line of the image is plotted with original filter size and with bilinearly upscaled values. The effect disappears if the low-resolution magnitude responses are upscaled instead of upscaling real and imaginary parts separately (Fig. 19(b)). The reason for this is that while the real and imaginary values are varying to a great degree between neighboring pixels the magnitudes are quite smooth, and therefore, upscaling creates no undesired effects.
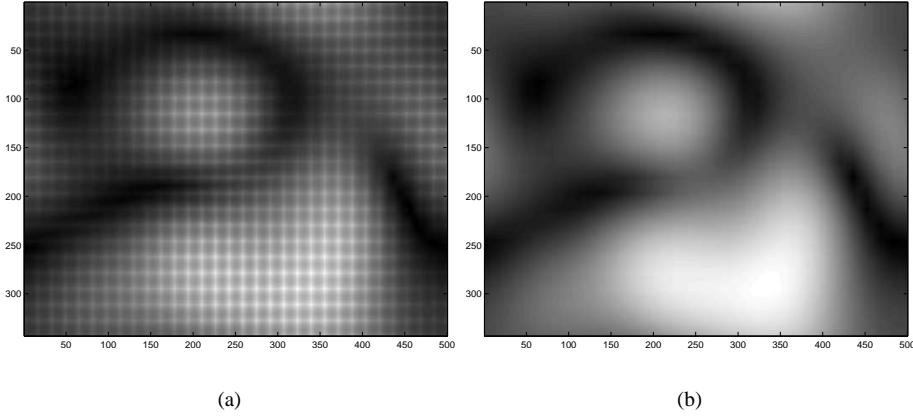

(a)   (b)

Figure 19: Filter responses from a filter at $f = 0.0177$ and orientation $135°$ with image downscaled to 1/16th; (a) Real and imaginary responses are upscaled to original resolution with bilinear filtering and magnitudes of responses are shown; (b) Magnitudes of responses are upscaled to original resolution with bilinear filtering.

The effect of downscaling during filtering was tested in Fig. 21. An image was filtered with a filter bank with four frequencies, the highest frequency being 0.03, and four orientations. The image was downscaled during filtering with scaling factors 1, 2, 4 and 8, and the responses were then upscaled back to the original resolution with nearest neighbor method (no interpolation) and with bilinear filtering. As was expected from the previous experiments, errors were smaller when absolute value responses were scaled as compared the the case where real and imaginary responses were upscaled separately.

While bilinear upscaling method leads to slightly smaller MSE, it cannot be recom-
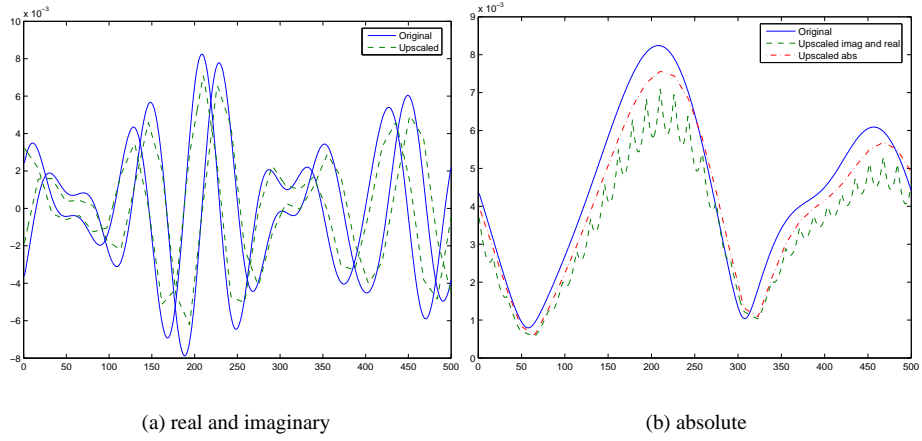
(a) real and imaginary

(b) absolute

Figure 20: Example of scaling error effect. (a) Real and imaginary parts of the filter response; (b) Magnitudes of the filter response.
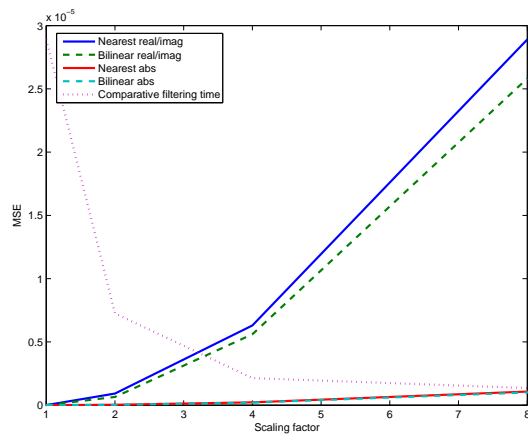

Figure 21: MSE between full size filtering and dowscaling procedure. Filtering time does not include upscaling.

mended because of previously mentioned anomalies (Fig. 19), which do not occur with nearest neighbor upscaling method. Filtering time is improving steadily when larger scaling factors are used, but it depends on the application whether loss of accuracy can be tolerated and to what degree. It must be noted that the downscaling of the image followed by upscaling the filter responses with nearest neighbor method leads to errors only because of low resolution of responses. For example, after scaling an image down by factor of 8, 64 pixels in the original image share the same filter response, while it is accurately correct for only one pixel.

6 Conclusions

Gabor features have shown to have beneficial properties in feature extraction for many computer vision tasks, but their computational complexity has prevented their use in practice. It was thus motivated to address the computational enhancements in this study.

Multi-resolution features are of special importance and it was pointed out how the computations can be significantly improved by utilizing effective filter envelopes, which reduce the computational complexity in the spatial domain and space complexity in the frequency domain. In addition, utilizing the highest required frequency for the image downscaling leads to a remarkable improvement for the both spatial and frequency domain filtering. Additionally, selection of filtering domain, either the spatial domain or the frequency domain, was considered for gaining maximum efficiency based on filter parameters and number of points to be filtered. Finally, the selection of filter bank parameters was defined so that all frequencies and orientations can be included.

With the provided results the Gabor filtering is finally enhanced to its optimal performance in the sequential programming and now the only available path is a parallel implementation, which on the other hand, would be natural for multi-resolution Gabor features. In the future use of the provided Gabor filter framework will be demonstrated in real-time computer vision tasks, especially in object tracking.

References

[1] A. Bernardino and J. Santos-Victor. A real-time Gabor primal sketch for visual attention. In *IBPRIA - 2nd Iberian Conference on Pattern Recognition and Image Analysis*, Estoril, Portugal, 2005.

[2] Alan Conrad Bovik, Marianna Clark, and Wilson S. Geisler. Multichannel texture analysis using localized spatial filters. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 12(1):55–73, January 1990.

[3] Ronald N. Bracewell. *The Fourier Transform and Its Applications, 3rd. edition*. McGraw-Hill, 2000.

[4] R.N. Braithwaite and B. Bhanu. Hierarchical gabor filters for object detection in infrared images. In *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, pages 628–631, 1994.

[5] P. Burt and E. Adelson. The Laplacian pyramid as a compact image code. *IEEE Transactions on Communications*, 31(4):532–540, 1983.

[6] John G. Daugman. High confidence visual recognition of persons by a test of statistical independence. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 15(11):1148–1161, 1993.

[7] M. Frigo and S.G. Johnson. The design and implementation of FFTW3. *This paper appears in: Proceedings of the IEEE*, 93(2):216–231, 2005.

[8] R.C. Gonzalez and R.E. Woods. *Digital Image Processing*. Prentice Hall, 2002.

[9] M. Hamouz, J. Kittler, J.-K. Kamarainen, P. Paalanen, H. Kälviäinen, and J. Matas. Feature-based affine-invariant detection and localization of faces. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 27(9):1480–1495, 2005.

[10] A. Hyvärinen, J. Karhunen, and E. Oja. *Independent Component Analysis*, chapter 2.7. John Wiley & Sons, Inc., 2001.

[11] J. Ilonen and J.-K. Kamarainen. Gabor filtering toolbox for Matlab. http://www.it.lut.fi/project/simplegabor/, 2005.

[12] J.-K. Kamarainen, V. Kyrki, and H. Kälviäinen. Invariance properties of gabor filter based features - overview and applications. *IEEE Transactions on Image Processing*, to be published.

[13] V. Kyrki, J.-K. Kamarainen, and H. Kälviäinen. Simple Gabor feature space for invariant object recognition. *Pattern Recognition Letters*, 25(3):311–318, 2004.

[14] M. Lades, J.C. Vorbrüggen, J. Buhmann, J. Lange, C. von der Malsburg, R.P. Würtz, and W. Konen. Distortion invariant object recognition in the dynamic link architecture. *IEEE Transactions on Computers*, 42(3):300–311, 1993.

[15] O. Nestares, R. Navarro, J. Portilla, and A. Tabernero. Efficient spatial-domain implementation of a multiscale image representation based on Gabor functions. *Journal of Electronic Imaging*, 7(1):166–173, 1998.

[16] Hyun Jin Park and Hyun Seung Yang. Invariant object detection based on evidence accumulation and Gabor features. *Pattern Recognition Letters*, 22:869–882, 2001.

[17] N. Ranganathan, R. Mehrotra, and K.R. Namuduri. An architecture to implement multiresolution. In *Proceedings of the International Conference on Acoustics, Speech, and Signal Processing (ICASSP)*, volume 2, pages 1157–1160, 1991.

[18] I.T. Young, L.J. van Vliet, and M. van Ginkel. Recursive Gabor filtering. *IEEE Transactions on Signal Processing*, 50(11):2798–2805, 2002.