Fast 2D Complex Gabor Filter With Kernel Decomposition

Jaeyoon Kim, Suhyuk Um, and Dongbo Min, *Senior Member, IEEE*

***Abstract*— 2D complex Gabor filtering has found numerous applications in the fields of computer vision and image processing. Especially, in some applications, it is often needed to compute 2D complex Gabor filter bank consisting of filtering outputs at multiple orientations and frequencies. Although several approaches for fast Gabor filtering have been proposed, they focus primarily on reducing the runtime for performing filtering once at specific orientation and frequency. To obtain the Gabor filter bank, the existing methods are repeatedly applied with respect to multiple orientations and frequencies. In this paper, we propose a novel approach that efficiently computes the 2D complex Gabor filter bank by reducing the computational redundancy that arises when performing filtering at multiple orientations and frequencies. The proposed method first decomposes the Gabor kernel to allow a fast convolution with the Gaussian kernel in a separable manner. This enables reducing the runtime of the Gabor filter bank by reusing intermediate results computed at a specific orientation. By extending this idea, we also propose a fast approach for 2D localized sliding discrete Fourier transform that uses the Gaussian kernel in order to lend spatial localization ability as in the Gabor filter. Experimental results demonstrate that the proposed method runs faster than the state-of-the-art methods, while maintaining similar filtering quality.**

***Index Terms*— 2-D complex Gabor filter, 2-D complex Gabor filter bank, 2-D localized sliding discrete Fourier transform (SDFT), kernel decomposition.**

I. Introduction

THANKS to the property of effectively extracting locally-varying structures from an image, 2-D complex Gabor filter has been widely used in a great variety of applications of computer vision and image processing, including texture analysis [1]–[3], face recognition [4]–[8], face expression recognition [9], [10] and fingerprint recognition [11]. It was known in [12], [13] that image analysis approaches based on the Gabor filter conceptually imitate the human visual system (HVS). The Gabor basis function defined for each pixel offers good spatial-frequency localization capability [14]. The 2-D complex Gabor filter is particularly useful for extracting a set of features at multiple orientations and frequencies from an image [15].

Performing 2-D complex Gabor filtering for all pixels over an entire image, however, often provokes a heavy computational cost. With the Gabor kernel defined at specific orientation and frequency, filtering is performed by moving a reference pixel one pixel at a time. The complex Gabor kernel hinders fast filtering in the context similar to edge-aware filters [16]–[18] widely used in computer vision applications.

To expedite 2-D complex Gabor filtering, several efforts have been made, for instance, by making use of fast Fourier transform (FFT), infinite impulse response (IIR) filters, or finite impulse response (FIR) filters [19]–[22]. It was shown in [19] that Gabor filtering for a 1-D signal of N samples can be performed with the same complexity as the FFT, $O(NlogN)$. In [20], separable FIR filters are applied to perform fast 2-D complex Gabor filtering by exploiting particular relationships between the Gabor parameters in a multiresolution pyramid. This method, however, works only for the particular setting of the Gabor parameters, e.g., scale of 2^i with an integer i. Young *et al.* [21] decomposed the Gabor filter with multiple IIR filters through z-transform, and then performed the recursive filtering in a manner similar to recursive Gaussian filtering [23]. To the best of our knowledge, the fastest algorithm is the work of Bernardino and Santos-Victor [22] that decomposes Gabor filtering into more efficient Gaussian filtering and sinusoidal modulations. It reduces the number of arithmetic operations by 39% compared to [21]. In [24], the 2-D Gabor filter is decomposed into three 1-D filters, and each filter is implemented on graphics processing units (GPUs). In [25], the Gabor filter with an anisotropic Gaussian kernel is accelerated by approximating the Gabor kernel into two separable 1-D convolutions, which can be efficiently implemented on GPUs. Note that the method employs an elongated Gaussian kernel and thus approximates it through more complicate operation such as singular value decomposition (SVD) [25].

These approaches aim to reduce the runtime required when performing 2-D complex Gabor filtering once at specific orientation and frequency. However, some computer vision applications require computing the 2-D complex *Gabor filter bank* consisting of filtering outputs at multiple orientations and frequencies in order to cope with geometric variation [3]–[5],

Manuscript received April 20, 2017; revised October 17, 2017 and November 24, 2017; accepted November 24, 2017. Date of publication December 14, 2017; date of current version January 12, 2018. This work was supported by the Basic Science Research Program through the National Research Foundation of Korea under Grant NRF-2015R1D1A1A01061143. The associate editor coordinating the review of this manuscript and approving it for publication was Dr. Alin M. Achim. *(Jaeyoon Kim and Suhyuk Um contributed equally to this work.) (Corresponding author: Dongbo Min.)*

J. Kim is with the Department of Computer Science, KAIST, Daejeon 34141, South Korea (e-mail: wodbs135@naver.com).

S. Um is with the Department of Computer Science and Engineering, Chungnam National University, Daejeon 34134, South Korea (e-mail: suhyuk1104@gmail.com).

D. Min is with the Department of Computer Science and Engineering, Ewha Womans University, Seoul 03760, South Korea (e-mail: dbmin99@gmail.com).

Color versions of one or more of the figures in this paper are available online at http://ieeexplore.ieee.org.

Digital Object Identifier 10.1109/TIP.2017.2783621

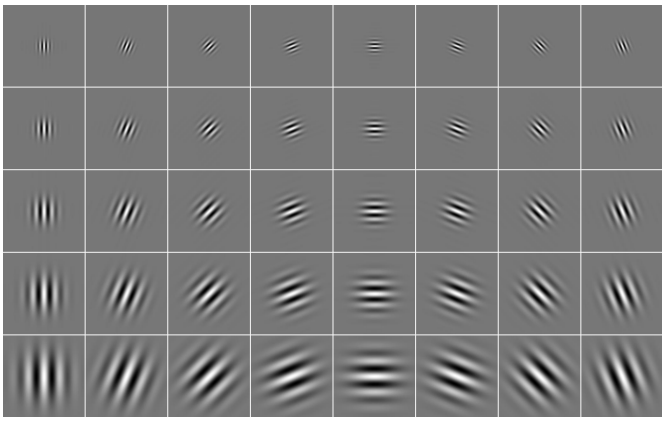


Fig. 1. Example of the 2-D complex Gabor filter bank with 40 coefficients (5 frequencies and 8 orientations). The coefficients are computed by $\omega = 2^{-(i+2)/2}$ $(i = 0, ..., 4)$, $\theta = k\pi/8$ $(k = 0, ..., 7)$ and $\sigma = 2\pi^2/\omega$ [4].

[7], [8], [15], [26]. For instance, face recognition approaches relying on the Gabor feature usually perform filtering at 8 orientations and 5 frequencies (totally, 40 Gabor feature maps) [4], [5], [7], [8], [26]. Fig. 1 shows the example of the Gabor filter kernels. To compute the Gabor filter bank, existing approaches [21], [22] repeatedly perform filtering for a given set of frequencies and orientations, disregarding the computational redundancy that exists in such repeated calculations.

In this paper, we propose a novel approach that efficiently computes the 2-D complex Gabor filter bank. The Gabor kernel is first decomposed using the trigonometric identities in order to perform a fast Gaussian convolution in a separable manner. This leads to a substantial reduction of the computational complexity when performing Gabor filtering at a set of orientations and frequencies. Specifically, intermediate results of Gabor filtering computed at a specific orientation are reused when performing filtering at its symmetric orientation. We will show that the proposed method runs faster when compared to state-of-the-art methods [21], [22], while maintaining similar filtering quality.

It is generally known that good spatial localization of the Gabor filter stems mainly from the use of the Gaussian kernel that determines an weight based on a spatial distance. We demonstrate that the fast computation of the 2-D *localized* sliding discrete Fourier transform (SDFT) using the Gaussian kernel is also possible using our kernel decomposition strategy. This lends spatial localization ability as in the Gabor filter to DFT results. Although numerous methods for fast 2-D SDFT have been proposed [27]–[29], they should use the box kernel within a sliding transform window. For instance, the relation between two successive 2-D DFT outputs is first derived using a circular shift property, and the 2-D DFT output at the current window is efficiently updated by linearly combining the 2-D DFT output at the previous window and one 1-D DFT result [29]. Note that the circular shift property holds only when the box kernel is used. Therefore, it is infeasible to use the existing 2-D SDFT methods [27]–[29] for calculating the 2-D *localized* DFT outputs using the Gaussian kernel. It should be noted that existing fast 2-D complex Gabor filters [21], [22] can be readily used to compute the 2-D localized SDFT, but our method still runs much faster due to the similar reason to the Gabor filter bank.

To sum up, our contributions can be summarized as follows.

- A new method is presented for efficiently computing the 2-D complex *Gabor filter bank*.
- The proposed method is extended into the 2-D *localized* SDFT.
- Extensive comparison with state-of-the-arts approaches is given in both analytic and experimental manners.

The rest of this paper is organized as follows. In Section II, we present the novel approach that efficiently computes the 2-D complex Gabor filter bank. In Section III, the proposed approach is extended to accelerate the 2-D localized SDFT. Section IV presents experimental results including runtime and filtering quality comparison with state-of-the-arts methods. Section V concludes this paper with some remarks.

II. Fast 2-D Complex Gabor Filter Bank

This section presents a new method that efficiently computes the 2-D complex Gabor filter bank consisting of filtering outputs at multiple orientations and frequencies. We introduce the Gabor kernel decomposition method and then show how it can be used for fast computation of the Gabor filter bank. Note that similar 1-D separable implementation was also used in [21] for fast Gabor filtering, but they do not consider the computational redundancy that exists in computing the Gabor filter bank.

For orientation θ and frequency $\omega = 2\pi/\lambda$ with wavelength λ, a 2-D complex Gabor filtering output $F_{\omega,\theta,\sigma}$ of a 2-D image f is written as

$$F_{\omega,\theta,\sigma}(x, y) = \sum_{k,l} f(k, l) C_{\omega,\theta}(x-k, y-l) G_{\sigma}(x-k, y-l) \tag{1}$$

where $G_{\sigma}(x, y)$ is 2-D Gaussian function with zero mean and the standard deviation of σ. Here, an isotropic Gaussian kernel with the same standard deviation for both x and y dimensions is used as in existing methods [21], [22], i.e., $G_{\sigma}(x, y) = S_{\sigma}(x)S_{\sigma}(y)$. The complex exponential function is defined as $C_{\omega,\theta}(x, y) = H_{\omega,\theta}(x)V_{\omega,\theta}(y)$, where $H_{\omega,\theta}(x) = e^{i\omega x \cos\theta}$ and $V_{\omega,\theta}(y) = e^{i\omega y \sin\theta}$.

A. Kernel Decomposition

Since $G_{\sigma}(x, y)$ and $C_{\omega,\theta}(x, y)$ are separable for x and y dimensions, (1) can be rewritten as

$$J_{\omega,\theta,\sigma}(x, y) = \sum_{k} f(k, y) H_{\omega,\theta}(x-k) S_{\sigma}(x-k), \tag{2}$$

$$F_{\omega,\theta,\sigma}(x, y) = \sum_{l} J_{\omega,\theta,\sigma}(x, l) V_{\omega,\theta}(y-l) S_{\sigma}(y-l), \tag{3}$$

$J_{\omega,\theta,\sigma}$ is first computed by performing 1-D horizontal Gabor filtering, and this is then used in 1-D vertical filtering for obtaining the final Gabor output $F_{\omega,\theta,\sigma}$.

1) 1-D Horizontal Gabor Filtering: We first present the efficient computation of $J_{\sigma,\omega,\theta}$ in (2) based on the kernel decomposition. We also omit y in $J_{\sigma,\omega,\theta}$ and f as the 1-D operation is repeated for $y = 1, ..., H$. Using the trigonometric identity, we can simply decompose (2) into two terms as

$$\mathcal{R}\{J_{\omega,\theta,\sigma}(x)\} = \cos(\omega_\theta^c x)\sum_k f_c(k)S_\sigma(x-k) + \sin(\omega_\theta^c x)\sum_k f_s(k)S_\sigma(x-k), \quad (4)$$

$$\mathcal{I}\{J_{\omega,\theta,\sigma}(x)\} = -\cos(\omega_\theta^c x)\sum_k f_s(k)S_\sigma(x-k) + \sin(\omega_\theta^c x)\sum_k f_c(k)S_\sigma(x-k), \quad (5)$$

where $\omega_\theta^c = \omega\cos\theta$, $f_c(k) = f(k)\cos(\omega_\theta^c k)$, and $f_s(k) = f(k)\sin(\omega_\theta^c k)$. $\mathcal{R}(J)$ and $\mathcal{I}(J)$ represent the real and imagery parts of J, respectively. (4) and (5) can be simply computed by applying the 1-D Gaussian smoothing to two modulated signals f_c and f_s. Here, we adopted the recursive Gaussian filtering proposed in [30], where the computational complexity per pixel is independent of the smoothing parameter σ.

2) 1-D Vertical Gabor Filtering: After $J(x, y)$ is computed using (4) and (5) for all $y = 1, .., H$, we perform 1-D Gabor filtering on the vertical direction using (3). Using the trigonometric identity, we decompose the real and imagery parts of F in (3) as follows:

$$\mathcal{R}\{F_{\omega,\theta,\sigma}(x,y)\} = \cos(\omega_\theta^s y)\left(f'_{cr}(x,y) + f'_{si}(x,y)\right) + \sin(\omega_\theta^s y)\left(f'_{sr}(x,y) - f'_{ci}(x,y)\right), \quad (6)$$

$$\mathcal{I}\{F_{\omega,\theta,\sigma}(x,y)\} = \sin(\omega_\theta^s y)\left(f'_{cr}(x,y) + f'_{si}(x,y)\right) - \cos(\omega_\theta^s y)\left(f'_{sr}(x,y) - f'_{ci}(x,y)\right), \quad (7)$$

where $\omega_\theta^s = \omega\sin\theta$. Here, f'_{cr}, f'_{sr}, f'_{ci}, and f'_{si} are filtering results convolved with 1-D Gaussian kernel S_σ as follows:

$$f'_{cr}(x,y) + f'_{si}(x,y) = \sum_l (f_{cr}(x,l) + f_{si}(x,l))S_\sigma(y-l),$$
$$f'_{sr}(x,y) - f'_{ci}(x,y) = \sum_l (f_{sr}(x,l) - f_{ci}(x,l))S_\sigma(y-l), \quad (8)$$

where the modulated signals f_{cr}, f_{sr}, f_{ci}, and f_{si} are defined as

$$f_{cr}(x,y) = \mathcal{R}\{J_{\omega,\theta,\sigma}(x,y)\}\cos(\omega_\theta^s y),$$
$$f_{sr}(x,y) = \mathcal{R}\{J_{\omega,\theta,\sigma}(x,y)\}\sin(\omega_\theta^s y),$$
$$f_{ci}(x,y) = \mathcal{I}\{J_{\omega,\theta,\sigma}(x,y)\}\cos(\omega_\theta^s y),$$
$$f_{si}(x,y) = \mathcal{I}\{J_{\omega,\theta,\sigma}(x,y)\}\sin(\omega_\theta^s y). \quad (9)$$

Like the horizontal filtering, two 1-D Gaussian convolutions are required in (6) and (7), i.e., $f'_{cr}(x,l) + f'_{si}(x,l)$ and $f'_{sr}(x,l) - f'_{ci}(x,l)$. In short, decomposing the complex exponential basis function $C_{\omega,\theta}$ enables us to apply the fast Gaussian filtering [30].

B. Fast Computation of 2-D Complex Gabor Filter Bank

Here, we show the kernel decomposition can be used to reduce the computational complexity when computing the 2-D complex Gabor filter bank (see Fig. 1). For a specific frequency ω, we aim at computing the Gabor filter bank at N orientations $\{\frac{\pi k}{N} | k = 0, ..., N-1\}$, and this is repeated for all frequencies. Fig. 2 (a) visualizes the log polar grid used in the Gabor filter bank with 5 frequencies and 8 orientations. For the simplicity of notation, we omit ω and σ in all equations. We assume that F_θ in (1) is computed using the proposed kernel decomposition technique and intermediate results are stored. $J_{\pi-\theta}$ and $F_{\pi-\theta}$ are then computed by recycling the intermediate results. Using $H_{\omega,\pi-\theta}(x) = H^*_{\omega,\theta}(x)$, where $*$ denotes complex conjugation, we obtain

$$J_{\pi-\theta}(x,y) = J^*_\theta(x,y). \quad (10)$$

The 1-D horizontal Gabor filtering result $J_{\pi-\theta}$ is complex conjugate to J_θ. Similarly, using $V_{\omega,\pi-\theta}(x) = V_{\omega,\theta}(x)$, the 1-D vertical Gabor filtering in $F_{\pi-\theta}$ is then expressed as

$$F_{\pi-\theta}(x,y) = \sum_l J^*_\theta(x,l)V_{\omega,\theta}(y-l)S_\sigma(y-l). \quad (11)$$

$F_{\pi-\theta}$ is obtained by applying the 1-D vertical Gabor filtering to the complex conjugate signal J^*_θ. Similar to (6) and (7), the following equations are derived:

$$\mathcal{R}\{F_{\pi-\theta}(x,y)\} = \cos(\omega_\theta^s y)\left(f'_{cr}(x,y) - f'_{si}(x,y)\right) + \sin(\omega_\theta^s y)\left(f'_{sr}(x,y) + f'_{ci}(x,y)\right), \quad (12)$$

$$\mathcal{I}\{F_{\pi-\theta}(x,y)\} = \sin(\omega_\theta^s y)\left(f'_{cr}(x,y) - f'_{si}(x,y)\right) - \cos(\omega_\theta^s y)\left(f'_{sr}(x,y) + f'_{ci}(x,y)\right) \quad (13)$$

The vertical filtering also requires two 1-D Gaussian convolutions. Algorithm 1 summarizes the proposed method for computing the 2-D complex Gabor filter bank. When a set of frequencies Ω and orientations Θ are given, we compute the Gabor filtering results at θ_k $(k = 0, ..., N-1)$ with the frequency ω_i being fixed. Different from existing approaches [21], [22] repeatedly applying the Gabor filter at all orientations, we consider the computational redundancy that exists on such repeated calculations. We will demonstrate

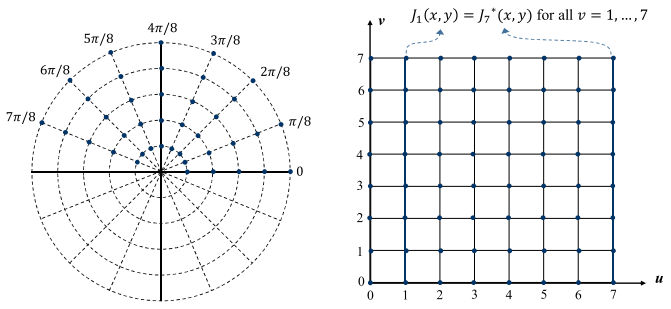


Fig. 2. Log polar grid of the 2-D complex Gabor filter bank and the rectangular grid of the 2-D localized SDFT. (a) 5 frequencies and 8 orientations, (b) 8×8 window ($M = 8$). In the log polar grid, two 1-D horizontal Gabor outputs are complex conjugate, i.e., $J_{\omega,\theta,\sigma} = J^*_{\omega,\pi-\theta,\sigma}$, when ω is fixed. In the 2-D SDFT, $J_u = J_{M-u}$ holds for $v = 0, ..., M-1$. These intermediate results can be reused in the computation of the Gabor filter bank and the 2-D localized SDFT. Note that though the number of orientations and M are set to even in (a) and (b), the proposed method is applicable regardless of whether they are even or odd.

Algorithm 1 Pseudo Code of the 2-D Complex Gabor Filter Bank

Input: input image f $(H \times W)$,
  a set of O scales $\Sigma = \{\sigma_i | i = 0, ..., O-1\}$,
  a set of O frequencies $\Omega = \{\omega_i | i = 0, ..., O-1\}$,
  a set of N orientations $\Theta = \{\theta_k | k = 0, ..., N-1\}$
Output: 2-D complex Gabor filter outputs for Ω and Θ

for $i = 0, ..., O-1$ **do** ▷ For all frequencies
  $\sigma_i = 2\pi^2/\omega_i, N_h = \lfloor N/2 \rfloor$
  for $k = 0, ..., N_h$ **do** ▷ For half of all orientations
    $\theta_k = \pi k/N$
    for $y = 1, ..., H$ **do**
      Perform 1-D Gaussian filtering of f_c, f_s
      Compute $J_{\omega_i,\theta_k,\sigma_i}(x, y)$ for all x in (4) and (5)
    end for
    for $x = 1, ..., W$ **do**
      Perform 1-D Gaussian filtering of $f_{cr} + f_{si}$, $f_{sr} - f_{ci}$ in (6) and (7)
      Compute $F_{\omega_i,\theta_k,\sigma_i}(x, y)$ for all y
    end for
  end for
  for $k = N_h + 1, ..., N-1$ **do** ▷ For remaining ori.
    $\theta_k = \pi k/N$
    $J_{\omega_i,\theta_k,\sigma_i}(x, y) = J^*_{\omega_i,\pi-\theta_k,\sigma_i}(x, y)$ for all x and y.
    for $x = 1, ..., W$ **do**
      Perform 1-D Gaussian filtering of $f_{cr} - f_{si}$, $f_{sr} + f_{ci}$ in (6) and (7)
      Compute $F_{\omega_i,\theta_k,\sigma_i}(x, y)$ for all y
    end for
  end for
end for

through both experimental and analytic comparisons that our method runs faster than existing methods [21], [22], when computing the Gabor filter bank.

III. 2-D Localized Sliding DFT

It is known that the Gabor filter offers good spatial localization ability thanks to the Gaussian kernel that determines an weight based on a spatial distance. Inspired by this, we present a new method that efficiently computes the 2-D *localized* SDFT using the proposed kernel decomposition technique. Note that different from the existing 2-D SDFT approaches [27]–[29] using the box kernel, the Gaussian kernel is used for computing the DFT at the sliding window. It is infeasible to apply the existing 2-D SDFT approaches [27]–[29] for calculating DFT outputs with the Gaussian kernel.

A. Kernel Decomposition in 2-D Localized SDFT

When the sliding window of $M \times M$ is used, we set the standard deviation σ of the Gaussian kernel by considering a cut-off range, e.g., $\lfloor M/2 \rfloor = 3\sigma$. We denote $F_{u,v}(x, y)$ by the

(a) (b) (c) (d)

Fig. 3. Some of images used in the experiment (USC-SIPI database [31]): (a) aerial image, (b) misc image, (c) texture image, and (d) sequence image.

$(u, v)^{th}$ bin of the $M \times M$ DFT at (x, y). The 2-D localized SDFT using the Gaussian kernel can be written as

$$F_{u,v}(x, y) = \sum_{m,n} f(m, n) C_{u,v}(\widehat{x} - m, \widehat{y} - n) G_\sigma(x - m, y - n) \quad (14)$$

where $\widehat{x} = x - \frac{M}{2}$ and $\widehat{y} = y - \frac{M}{2}$. Similar to the Gabor filter bank, the DFT is performed for $u, v = 0, ..., M-1$. The complex exponential function $C_{u,v}(m, n)$ at the $(u, v)^{th}$ bin is defined as $C_{u,v}(x, y) = H_u(x)V_v(y)$, where $H_u(x) = e^{i\omega_0 ux}$ and $V_v(y) = e^{i\omega_0 vy}$. Here, $\omega_0 = \frac{2\pi}{M}$ represents a base frequency. Note that in (14), slightly different notations from the conventional SDFT methods [27]–[29] are used to keep them consistent with the Gabor filter of (1). When $G_\sigma(x, y) = 1$, (14) becomes identical to the conventional 2-D SDFT methods [27]–[29]. The Gaussian window of $M \times M$ is used here for simplicity of notations, but the 2-D localized SDFT using $M_y \times M_x$ window $(M_y \neq M_x)$ is also easily derived.

Using the separable property of $G_\sigma(x, y)$ and $C_{u,v}(x, y)$, (14) can be written as

$$J_u(x, y) = \sum_m f(m, y) H_u(\widehat{x} - m) S_\sigma(x - m), \quad (15)$$

$$F_{u,v}(x, y) = \sum_n J_u(x, n) V_v(\widehat{y} - n) S_\sigma(y - n). \quad (16)$$

Using the kernel decomposition, the 1-D horizontal localized SDFT is performed as follows:

$$\mathcal{R}\{J_u(x)\} = \cos(\omega_0\, u\widehat{x}) \sum_m f_c(m) S_\sigma(x - m) + \sin(\omega_0\, u\widehat{x}) \sum_m f_s(m) S_\sigma(x - m), \quad (17)$$

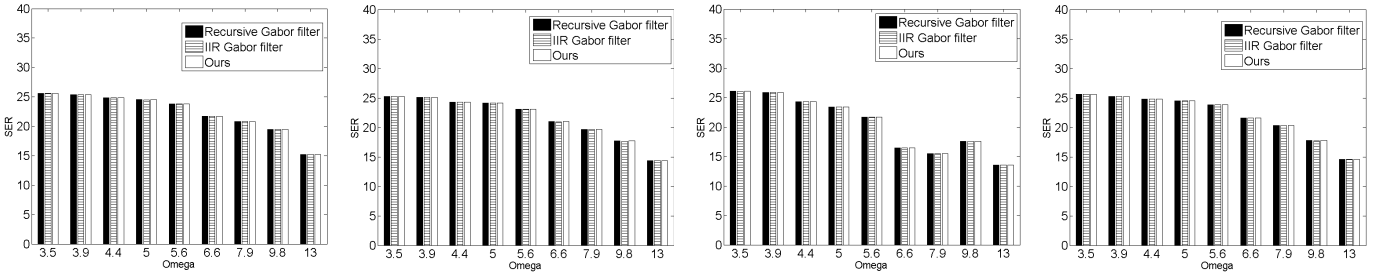
$$\mathcal{I}\{J_u(x)\} = -\cos(\omega_0 u\widehat{x}) \sum_m f_s(m) S_\sigma(x - m) + \sin(\omega_0 u\widehat{x}) \sum_m f_c(m) S_\sigma(x - m), \quad (18)$$

where $f_c(m) = f(m)cos(\omega_0\, um)$, $f_s(m) = f(m)sin(\omega_0\, um)$.

The 1-D vertical localized SDFT is performed similar to the Gabor filter:

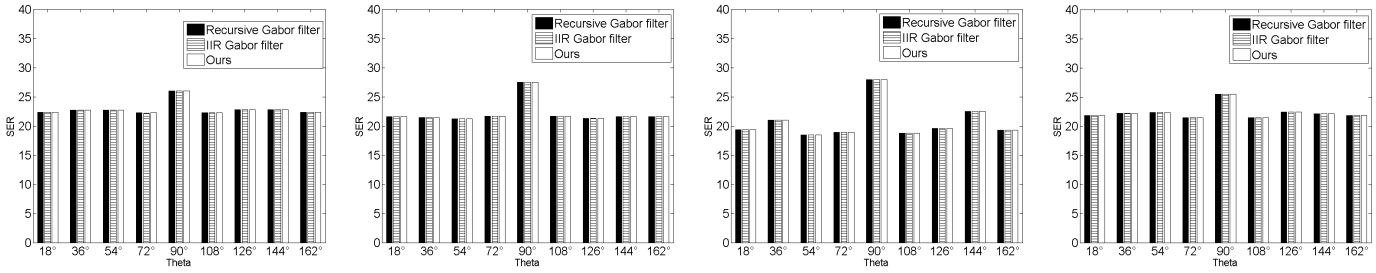
$$\mathcal{R}\{F_{u,v}(x, y)\} = \cos(\omega_0 v\widehat{y}) \left(f'_{cr}(x, y) + f'_{si}(x, y)\right) + \sin(\omega_0 v\widehat{y}) \left(f'_{sr}(x, y) - f'_{ci}(x, y)\right), \quad (19)$$

$$\mathcal{I}\{F_{u,v}(x, y)\} = \sin(\omega_0 v\widehat{y}) \left(f'_{cr}(x, y) + f'_{si}(x, y)\right) - \cos(\omega_0 v\widehat{y}) \left(f'_{sr}(x, y) - f'_{ci}(x, y)\right), \quad (20)$$



(a) (b) (c) (d)

Fig. 4. Objective comparison using the imagery parts of 2-D complex Gabor filtering outputs with the varying frequency ω when $\theta = \pi/3$. We compared the average SER values of three methods, the recursive Gabor filter [21], IIR Gabor filter [22], and our method, for four datasets: (a) aerial, (b) miscellaneous, (c) sequences, and (d) textures.



(a) (b) (c) (d)

Fig. 5. Objective comparison using the imagery parts of 2-D Gabor filtering outputs with the varying orientation θ when $\omega = 13$. Similar to Fig. 4, the average SER values were measured using the recursive Gabor filter [21], IIR Gabor filter [22], and our method. (a) Aerial. (b) Misc. (c) Sequences. (d) Textures.

where $f'_{cr} + f'_{si}$ and $f'_{sr} - f'_{ci}$ are defined in a manner similar to (8). J_u and $F_{u,\upsilon}$ are computed by applying 1-D Gaussian smoothing twice, respectively.

B. Exploring Computational Redundancy on (u, υ)

The 2-D localized SDFT consists of a set of DFT outputs for $u, \upsilon = 0, ..., M-1$. Considering the conjugate symmetry property of the DFT ($F_{M-u,M-\upsilon} = F^*_{u,\upsilon}$), we compute the DFT outputs $F_{u,\upsilon}$ only for $u = 0, ..., M-1$ and $\upsilon = 0, ..., \lfloor M/2 \rfloor$, and then simply obtain remaining DFT outputs (for $u = 0, ..., M-1$ and $\upsilon = \lfloor M/2 \rfloor + 1, ..., M-1$) using the complex conjugation. Similar to the Gabor filter bank, the 1-D DFT J_{M-u} is complex conjugate to J_u as follows:

$$\begin{aligned} J_{M-u}(x, y) &= \sum_m f(m, y) H_{M-u}(\widehat{x} - m) S_\sigma(x - m), \\ &= \sum_m f(m, y) H^*_u(\widehat{x} - m) S_\sigma(x - m), \\ &= J^*_u(x, y) \end{aligned} \tag{21}$$

The 1-D vertical SDFT result $F_{M-u,\upsilon}$ is then obtained as

$$F_{M-u,\upsilon}(x, y) = \sum_n J^*_u(x, n) V_\upsilon(\widehat{y} - n) S_\sigma(y - n). \tag{22}$$

As in the Gabor filter bank, (22) can be computed by performing 1-D Gaussian filtering twice.

Fig. 2 (b) shows the example of the regular grid ($M = 8$) used in the 2-D SDFT. There exists an additional computational redundancy when performing 2-D SDFT on the regular grid. For a specific u, the 1-D horizontal filtering results $J_u(x, y)$ remain unchanged for $\upsilon = 0, ..., \lfloor M/2 \rfloor$, and also $J_{M-u}(x, y) = J^*_u(x, y)$ is simply computed. These results are used as inputs for the 1-D vertical SDFT.

Algorithm 2 shows the overall process of computing the 2-D localized SDFT. Here, we explain the method with a non-square window of $M_y \times M_x$ ($M_y \geq M_x$) for a generalized description. When $M_y \geq M_x$, 1-D horizontal and vertical filters are performed at lines $4-9$ and $13-18$, respectively, and *vice versa* in order to reduce the runtime. This is because the filtering order affects the 1-D SDFT complexity at lines $4-9$, while having no effect on the complexity of lines $13-18$ when $M_y \neq M_x$. Algorithm 2 can be simply modified when $M_y < M_x$, i.e., by performing 1-D vertical and horizontal filtering at lines $4-9$ and $13-18$, respectively. The number of arithmetic operations is also reported in Table IV.

To obtain $M_y \times M_x$ ($M_y \geq M_x$) DFT outputs in Algorithm 2, we first obtain $J_u(x, y)$ for $u = 0, ..., \lfloor M_x/2 \rfloor$ using (17) and (18), and then simply calculate $J_u(x, y)$ for $u = \lfloor M_x/2 \rfloor + 1, ..., M_x - 1$ using (21). $J_u(x, y)$ is then used to obtain $F_{u,\upsilon}(x, y)$ by performing 1-D vertical filtering. Thus, the horizontal filtering $J_u(x, y)$ is performed only for $u = 0, ..., \lfloor M_x/2 \rfloor$, while the vertical filtering $F_{u,\upsilon}(x, y)$ is done for $u = 0, ..., M_x - 1$ and $\upsilon = 0, ..., \lfloor M_y/2 \rfloor$. Finally, remaining DFT outputs (for $u = 0, ..., M_x - 1$ and $\upsilon = \lfloor M_y/2 \rfloor + 1, ..., M-1$) are obtained using the conjugate symmetry property at lines $19-21$.

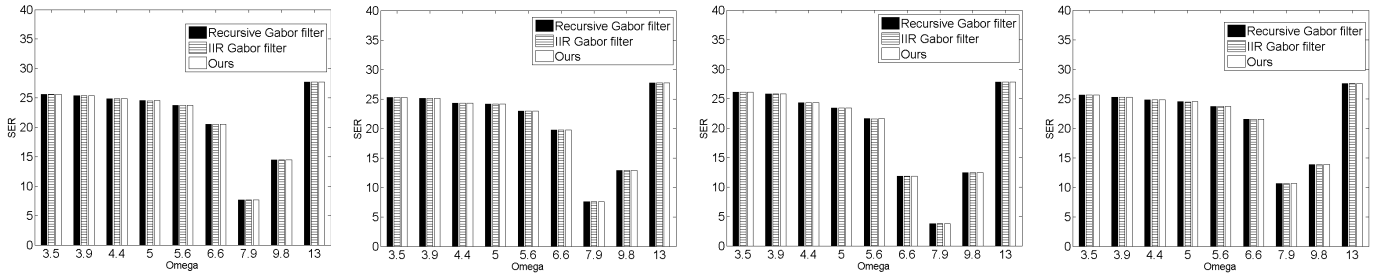


Fig. 6. Objective comparison using the real parts of 2-D complex Gabor filtering outputs with the varying frequency ω when $\theta = \pi/3$. The SER values were measured in a manner similar to Fig. 4. (a) Aerial. (b) Misc. (c) Sequences. (d) Textures.

Algorithm 2 Pseudo Code of the 2-D Localized SDFT

Input: input image f $(H \times W)$, scale σ, kernel size $M_y \times M_x$ $(M_y \geq M_x)$
Output: SDFT outputs at $u = 0, ..., M_x - 1$ and $v = 0, ..., M_y - 1$

$M_{xh} = \lfloor M_x/2 \rfloor$, $M_{yh} = \lfloor M_y/2 \rfloor$
for $u = 0, ..., M_{xh}$ **do**
  for $y = 1, ..., H$ **do** ▷ 1-D horizontal SDFT
    Perform 1-D Gaussian filtering of f_c, f_s in (17) and (18).
    Compute $J_u(x, y)$ for all x.
  end for
end for
for $u = M_{xh} + 1, ..., M_x - 1$ **do**
  $J_{M_x - u}(x, y) = J_u^*(x, y)$ for all x and y.
end for

for $u = 0, ..., M_x - 1$, $v = 0, ..., M_{yh}$ **do**
  for $x = 1, ..., W$ **do** ▷ 1-D vertical SDFT
    Perform 1-D Gaussian filtering of $f_{cr} + f_{si}$, $f_{sr} - f_{ci}$ in (19) and (20).
    Compute $F_{u,v}(x, y)$ for all y.
  end for
end for

for $u = 0, ..., M_x - 1$, $v = M_{yh} + 1, ..., M_y - 1$ **do**
  $F_{u,v}(x, y) = F^*_{M_x - u, M_y - v}(x, y)$
end for

IV. Experimental Results

We compared the proposed method with state-of-the-arts methods [21], [22] for fast Gabor filtering in terms of both computational efficiency and filtering quality. For a fair comparison, we implemented the two methods [21], [22] with a similar degree of code optimization. All the codes including ours will be publicly available later for both the 2-D complex Gabor filter bank and the 2-D localized SDFT.

A. Computational Complexity Comparison

We first compared the runtime when computing the 2-D complex Gabor filter bank. As our method focuses on reducing the computational redundancy on the repeated application of the Gabor filter at multiple orientations, we compared only the runtime of the Gabor filter bank. Additionally, the runtime was analyzed by counting the number of arithmetic operations such as addition and multiplication. The runtime of the 2-D localized SDFT was also measured in both experimental and analytic manners. The existing fast Gabor filters [21], [22] were applied to compute the DFT outputs for all frequency bins of the 2-D localized SDFT. Note that conventional 2-D SDFT approaches using the box kernel [27]–[29] were not compared, since they are not capable of computing the 2-D localized DFT outputs.

Table I compares the runtime of the 2-D complex Gabor filter bank. As summarized in Algorithm 1, our method

TABLE I

Runtime Comparison (Millisecond) of the 2-D Complex Gabor Filter Bank. We Measured the Runtime When Computing the 2-D Complex Gabor Filter Bank for Multiple Orientations at a Specific Frequency. The Set of N Orientations Θ is Defined as $\{\theta_k = \frac{k\pi}{N} | k = 1, ..., N-1\}$. The Size of the Input Image is 1024×1024

N	**Recursive Gabor fil.** [21]	**IIR Gabor filter** [22]	**Ours**
8	608	500	359
14	1039	852	586
20	1518	1230	842
26	1972	1597	1079
32	2421	1971	1314

TABLE II

Computational Complexity Comparison of the 2-D Complex Gabor Filter Bank. Similar to Table I, When Computing the 2-D Complex Gabor Filter Bank for N Orientations at a Specific Frequency, We Count the Number of Multiplications R_M and Additions R_A Per Pixel, Respectively

Algorithm	Operation	The number of orientations N					
		8	14	20	26	30	N
[21]	R_M	416	728	1040	1352	1560	$52N$
	R_A	376	658	940	1222	1410	$47N$
[22]	R_M	272	476	680	884	1020	$34N$
	R_A	208	364	520	676	780	$26N$
Ours	R_M	240	420	600	780	900	$30N$
	R_A	176	308	440	572	660	$22N$

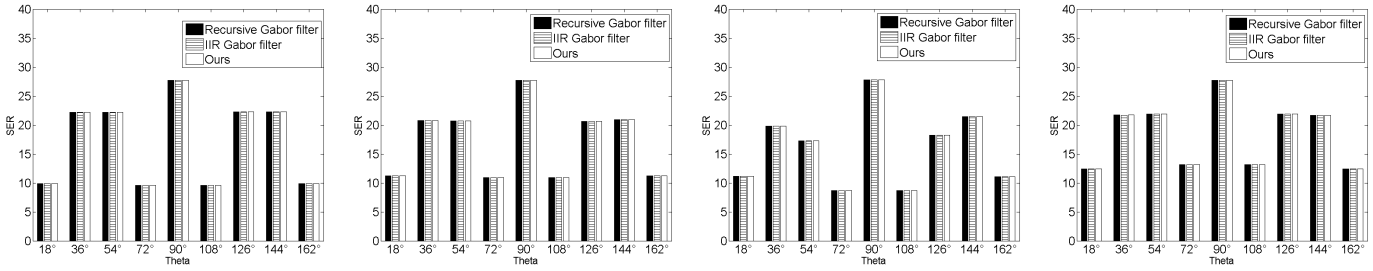


Fig. 7. Objective comparison using the real parts of 2-D complex Gabor filtering outputs with the varying orientation θ when $\omega = 13$. The SER values were measured in a manner similar to Fig. 5. (a) Aerial. (b) Misc. (c) Sequences. (d) Textures.

TABLE III

RUNTIME COMPARISON (MILLISECOND) OF THE 2-D LOCALIZED SDFT. THE WINDOW SIZE FOR DFT IS $M \times M$ WHERE $\lfloor M/2 \rfloor = 3\sigma$ IS SET WITH THE STANDARD DEVIATION σ OF THE GAUSSIAN KERNEL. WE ALSO COMPARED THE RUNTIME WITH TWO EXISTING METHODS [21], [22] BY REPEATEDLY APPLYING THEM WHEN COMPUTING $F_{u,v}$ FOR $u, v = 0, ..., M-1$. THE SIZE OF THE INPUT IMAGE IS 250×234

$M \times M$	**Recursive Gabor fil.** [21]	**IIR Gabor filter** [22]	**Ours**
8×8	45	101	40
10×10	67	159	57
12×12	94	228	75
14×14	127	317	101
16×16	163	421	125

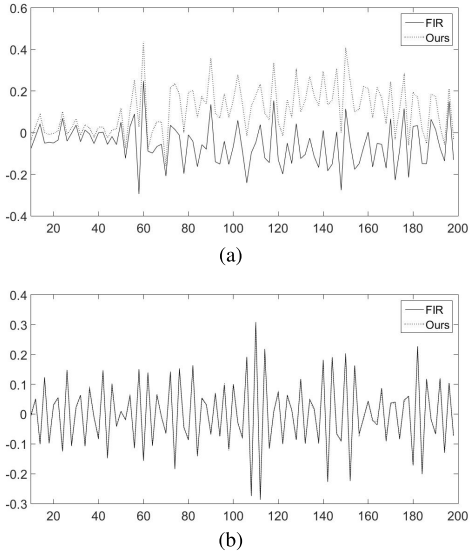


Fig. 8. 1-D profiles of 2-D complex Gabor filtering results: (a) the real part at $\omega = 7.9$ and $\theta = \pi/3$ when $SER = 10.57$, (b) the real part at $\omega = 3.5$ and $\theta = \pi/3$ when $SER = 25.61$.

can be applied to each frequency. Thus, we measured the runtime for N orientations when a specific frequency ω is given. The set of orientations Θ is defined as $\{\theta_k = \frac{k\pi}{N} | k = 0, ..., N-1\}$. In the existing fast Gabor filters [21], [22], there is no consideration of the computational redundancy that occurs when computing the Gabor filtering outputs at multiple orientations. The fast Gabor filter using IIR approximation [22] is computationally lighter than the recursive Gabor filter [21], but our method runs faster than the two methods. In Table II, we compared the number of arithmetic operations at N orientations and a single frequency ω, in the manner similar to Table I. We count the number of multiplications R_M and additions R_A per pixel, respectively. Considering R_M and R_A of the three approaches, the runtime results in Table I are in agreement.

Table III shows the runtime comparison of the 2-D localized SDFT. It requires computing all 2-D DFT outputs for $u, v = 0, ..., M-1$, when $M \times M$ window is used. Note that the conjugate symmetry property, i.e., $F_{u,v} = F^*_{M-u,M-v}$ was used for all methods for a fair comparison (see Algorithm 2). It is clearly shown that our method runs much faster than the two methods. Interestingly, our runtime gain against the IIR Gabor filter [22] becomes higher, when compared to the Gabor filter bank in Table I. This is because the 1-D horizontal DFT output J can be reused for $v = 0, ...M-1$ in the rectangular grid of Fig. 2 and it is also shared for both $M-u$ and u (see Algorithm 2) Namely, the ratio of shared computations increases in the 2-D localized SDFT, compared to the Gabor filter bank.

In Table III, we also found that the IIR Gabor filter [22] becomes slower than the recursive Gabor filter [21] when computing the 2-D localized SDFT, while the former runs faster than the latter in the Gabor filter bank (compare Table I and Table III). The IIR Gabor filter [22] decomposes the Gabor kernel into the complex sinusoidal modulation and the Gaussian kernel, and then performs Gaussian smoothing with 2-D modulated signals. Contrarily, the recursive Gabor filter [21] performs filtering in a separable manner, and thus we implement the 2-D localized SDFT using [21] such that it can reuse 1-D intermediate results, resulting in the faster runtime than [22]. In Table IV, we count the number of multiplications R_M and additions R_A, which is consistent with the runtime results in Table III. Here, we also count R_M and R_A when the non-square window of $M_y \times M_x$ ($M_y \neq M_x$) is used.

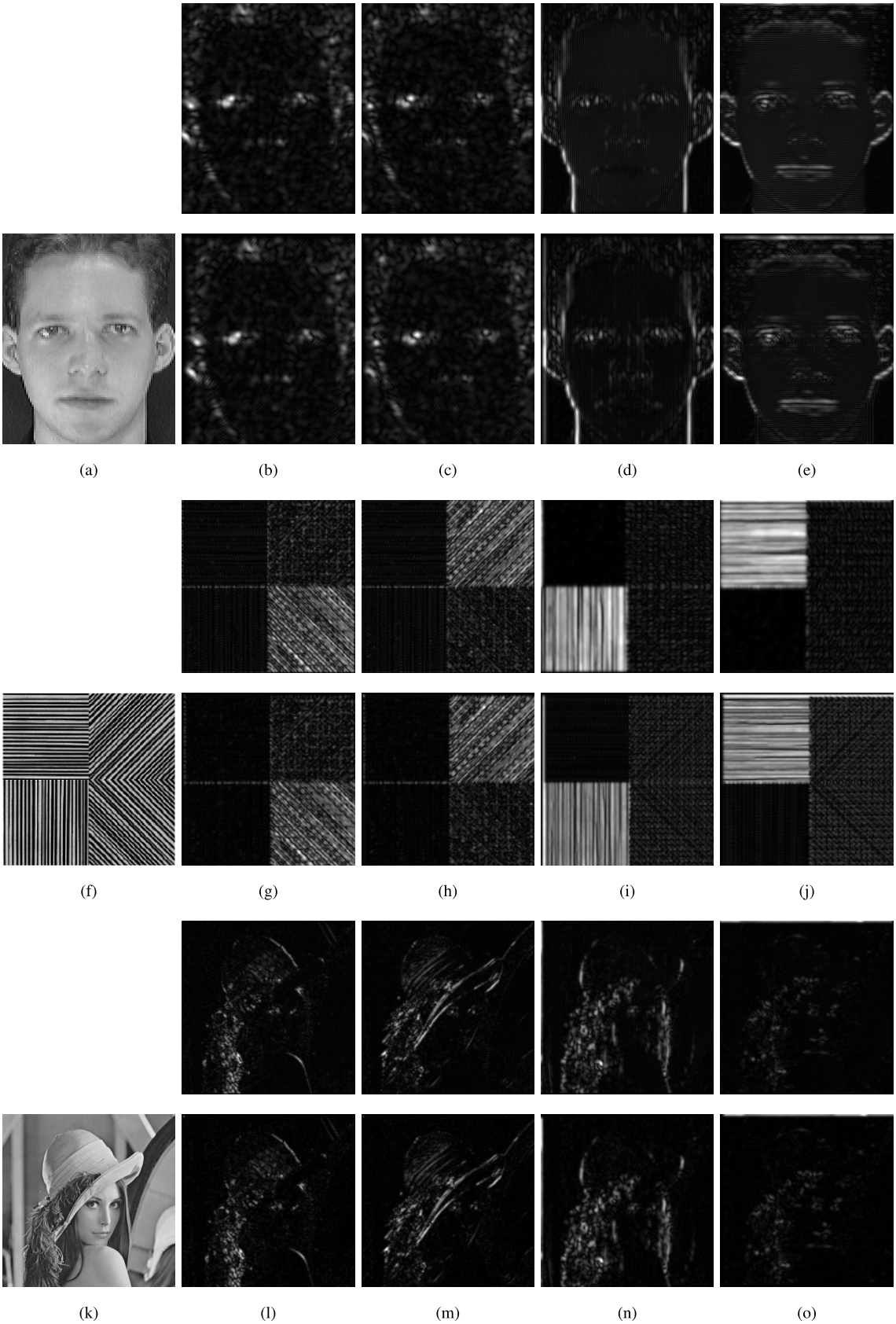


Fig. 9. 2-D complex Gabor filter bank outputs computed by our method and FFT based method for 'Face', 'Texture', and 'Lena' images. We visualize them with an absolute magnitude since the filtering results are in a complex form. The first and second rows of each image represent our results and FFT based results, respectively. $\lambda = \sigma/\pi$ depends on σ. (a) Face. (b) $\sigma = 3, \theta = \pi/4$. (c) $\sigma = 3, \theta = 3\pi/4$. (d) $\sigma = 2, \theta = 0$. (e) $\sigma = 2, \theta = \pi/2$. (f) Texture. (g) $\sigma = 2.7, \theta = \pi/4$. (h) $\sigma = 2.7, \theta = 3\pi/4$. (i) $\sigma = 4, \theta = 0$. (j) $\sigma = 4, \theta = \pi/2$. (k) Lena. (l) $\sigma = 2.7, \theta = \pi/4$. (m) $\sigma = 2.7, \theta = 3\pi/4$. (n) $\sigma = 4, \theta = 0$. (o) $\sigma = 4, \theta = \pi/2$.

TABLE IV

COMPUTATIONAL COMPLEXITY COMPARISON OF THE 2-D LOCALIZED SDFT. SIMILAR TO TABLE III, WE COMPARED WITH TWO EXISTING METHODS [21], [22]. THE WINDOW SIZE OF DFT IS $M \times M$. WE COUNT THE NUMBER OF MULTIPLICATIONS R_M AND ADDITIONS R_A PER PIXEL REQUIRED TO COMPUTE THE 2-D DFT $F_{u,v}$ FOR $u, v = 0, ..., M-1$. WE ALSO COUNT R_M AND R_A WHEN A NON-SQUARE WINDOW OF $M_y \times M_x$ ($M_y \neq M_x$) IS USED

Algorithm	Operation	Kernel size						
		1×1	2×2	4×4	8×8	16×16	$M_y \times M_x$ ($M_y \geq M_x$)	$M_y \times M_x$ ($M_y < M_x$)
Recursive Gabor fil. [21]	R_M	26	78	260	936	3536	$13M_xM_y + 13M_x$	$13M_xM_y + 13M_y$
	R_A	23.5	71	238	860	3256	$12M_xM_y + 11.5M_x$	$12M_xM_y + 11.5M_y$
IIR Gabor filter [22]	R_M	34	136	544	2179	8704	$34M_xM_y$	$34M_xM_x$
	R_A	26	104	416	1664	6656	$26M_xM_y$	$26M_xM_y$
Ours	R_M	18	54	180	684	2448	$9M_xM_y + 9M_x$	$9M_xM_y + 9M_y$
	R_A	14.5	44	148	536	2030	$7.5M_xM_y + 7M_x$	$7.5M_xM_y + 7M_y$

B. Filtering Quality Comparison

All the fast Gabor filtering methods including ours produce approximated results, as they count on the recursive Gaussian filter using IIR approximation [23], [30]. The IIR Gaussian filter runs fast at the cost of a filtering quality loss. It was reported in [23], [30] that the quality loss is negligible when using the standard deviation within an appropriate range. We compared the filtering quality with two fast Gabor filtering approaches [21], [22]. The filtering quality was measured for the 2-D complex Gabor filter bank only, as the 2-D localized SDFT tends to show similar filtering behaviors.

We used input images from the USC-SIPI database [31] which consists of four different classes of images: aerial images, miscellaneous images, sequence images, and texture images. Fig. 3 shows some of the sample images. The Gabor filtering result does not range from 0 to 255, different from an image. Thus, instead of the peak signal-to-noise ratio (PSNR) widely used in an image quality assessment, we computed the signal-to-error ratio (SER), following [22]:

$$SER[dB] = 10\log_{10}\frac{\sum_{x,y}\left(\mathcal{R}\{F(x,y)\}\right)^2}{\sum_{x,y}\left(\mathcal{R}\{F(x,y)\} - \mathcal{R}\{F_t(x,y)\}\right)^2},$$

where F and F_t are the results obtained using the fast method and the *lossless* FIR Gabor filter in (1), respectively. $\mathcal{R}(F)$ represents the real part of F. The SER can also be measured with the imagery part $\mathcal{I}(F)$. We computed the approximation error for the frequency $\omega \in \{3.5, 3.9, ..., 9.8, 13\}$ and the orientation $\theta \in \{18°, 36°, ..., 162°\}$.

Fig. 4 and 5 compare the objective quality of the Gabor filtering results. We measured the average SER values of the imagery parts with respect to the varying frequency ω and orientation θ for four datasets: aerial, miscellaneous, sequences, and textures images. The average SER values are all similar to for the three methods: the recursive Gabor filter [21], IIR Gabor filter [22], and ours. Four different classes of images did not show significantly different tendency in terms of the filtering quality. Fig. 6 and 7 shows the SER values measured using the real parts. Interestingly, the average SER values of the real parts at some frequencies and orientations become lower. It was explained in [22] that the difference between DC values of the lossless FIR and fast filters happens to become larger at these ranges. In Fig. 8, we plotted 1-D profiles using the real parts of Gabor filtering results for two cases with low and high SER values. The horizontal and vertical axes represent the pixel location and the real part value of the Gabor filter, respectively. In the case with the low SER value, we found that an overall tendency is somehow preserved with some offsets.

Fig. 9 shows the Gabor filter bank results of our method and FFT based method for 'Face', 'Texture', and 'Lena' images. Note that the filtering results of the FFT based method are identical to those of the lossless FIR Gabor filter in (1). The absolute magnitude of the complex result was used for visualization. The first and second rows of each image represent our results and FFT based results, respectively. We found the subjective quality of two methods to be very similar. Exceptionally, our results seem to be a bit blurred at the highly textured regions of Fig. 9 (i) and (j). This may be due to the IIR approximation used in the Gaussian filtering. Similar filtering results were also observed in the existing fast Gabor filters [21], [22].

V. CONCLUSION

We have presented a new method for fast computation of the 2-D complex Gabor filter bank at multiple orientations and frequencies. The proposed method achieved a substantial runtime gain by reducing the computational redundancy that arises when computing the Gabor filter bank. Additionally, this method was extended into the 2-D localized SDFT that uses the Gaussian kernel. The runtime gain was verified in both analytic and experimental manners. We showed that the proposed method maintains a similar level of filtering quality when compared to state-of-the-arts approaches for fast Gabor filtering, but it runs much faster. We believe that the proposed method for the fast Gabor filter bank is crucial to various computer vision tasks that require a low complexity.

The 2-D localized SDFT is expected to provide more useful information than the conventional 2-D SDFT approaches using the simple box kernel. We will continue to study the effectiveness of the 2-D localized SDFT in several computer vision applications as future work. Another interesting topic is the fast implementation of the log-Gabor filter [32] that resolves the non-zero DC value problem of the original Gabor filter. Note that our method is applicable only when the kernel is separable in x and y dimensions. Thus, it is not feasible to directly apply the proposed method to the log-Gabor filter using non-separable kernels. Nevertheless, its fast

implementation would be interesting, e.g., using separable approximation of the log-Gabor filter kernel.

REFERENCES

[1] T. P. Weldon, W. E. Higgins, and D. F. Dunn, "Efficient Gabor filter design for texture segmentation," *Pattern Recognit.*, vol. 29, no. 12, pp. 2005–2015, 1996.

[2] F. Bianconi and A. Fernández, "Evaluation of the effects of Gabor filter parameters on texture classification," *Pattern Recognit.*, vol. 40, no. 12, pp. 3325–3335, 2007.

[3] C. Li, G. Duan, and F. Zhong, "Rotation invariant texture retrieval considering the scale dependence of Gabor wavelet," *IEEE Trans. Image Process.*, vol. 24, no. 8, pp. 2344–2354, Aug. 2015.

[4] L. Wiskott, J.-M. Fellous, N. Krüger, and C. von der Malsburg, "Face recognition by elastic bunch graph matching," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 19, no. 7, pp. 775–779, Jul. 1997.

[5] C. Liu and H. Wechsler, "Independent component analysis of Gabor features for face recognition," *IEEE Trans. Neural Netw.*, vol. 14, no. 4, pp. 919–928, Jul. 2003.

[6] L. Shen and L. Bai, "A review on Gabor wavelets for face recognition," *Pattern Anal. Appl.*, vol. 9, nos. 2–3, pp. 273–292, 2006.

[7] Z. Lei, S. Liao, R. He, M. Pietikäinen, and S. Z. Li, "Gabor volume based local binary pattern for face representation and recognition," in *Proc. 8th IEEE Int. Conf. Automat. Face Gesture Recognit. (FG)*, Amsterdam, The Netherlands, Sep. 2008, pp. 1–6.

[8] Y. Cheng, Z. Jin, H. Chen, Y. Zhang, and X. Yin, "A fast and robust face recognition approach combining Gabor learned dictionaries and collaborative representation," *Int. J. Mach. Learn. Cybern.*, vol. 7, no. 1, pp. 47–52, 2016.

[9] W. Gu, C. Xiang, Y. V. Venkatesh, D. Huang, and H. Lin, "Facial expression recognition using radial encoding of local Gabor features and classifier synthesis," *Pattern Recognit.*, vol. 45, no. 1, pp. 80–91, 2012.

[10] M. K. Mandal and A. Awasthi, *Understanding Facial Expressions in Communication*. India: Springer, 2015.

[11] H. Kasban, "Fingerprints verification based on their spectrum," *Neurocomputing*, vol. 171, pp. 910–920, Jan. 2016.

[12] L. Shen, L. Bai, and M. Fairhurst, "Gabor wavelets and general discriminant analysis for face identification and verification," *Image Vis. Comput.*, vol. 25, no. 5, pp. 553–563, 2007.

[13] L. Xu, W. Lin, and C.-C. J. Kuo, *Visual Quality Assessment by Machine Learning*. Singapore: Springer, 2015.

[14] I. Daubechies, "The wavelet transform, time-frequency localization and signal analysis," *IEEE Trans. Inf. Theory*, vol. 36, no. 5, pp. 961–1005, Sep. 1990.

[15] J.-K. Kamarainen, V. Kyrki, and H. Kälviäinen, "Invariance properties of Gabor filter-based features-overview and applications," *IEEE Trans. Image Process.*, vol. 15, no. 5, pp. 1088–1099, May 2006.

[16] K. He, J. Sun, and X. Tang, "Guided image filtering," in *Computer Vision—ECCV* (Lecture Notes in Computer Science), vol. 6311. Berlin, Germany: Springer, 2010, pp. 1–14.

[17] E. S. L. Gastal and M. M. Oliveira, "Domain transform for edge-aware image and video processing," *ACM Trans. Graph.*, vol. 30, no. 4, p. 69, 2011.

[18] D. Min, S. Choi, J. Lu, B. Ham, K. Sohn, and M. N. Do, "Fast Global image smoothing based on weighted least squares," *IEEE Trans. Image Process.*, vol. 23, no. 12, pp. 5638–5653, Dec. 2014.

[19] S. Qiu, F. Zhou, and P. E. Crandall, "Discrete Gabor transforms with complexity O *(NlogN)*," *Signal Process.*, vol. 77, no. 2, pp. 159–170, 1999.

[20] O. Nestares, R. F. Navarro, J. Portilla, and A. Tabernero, "Efficient spatial-domain implementation of a multiscale image representation based on Gabor functions," *J. Electron. Imag.*, vol. 7, no. 1, pp. 166–173, 1998.

[21] I. T. Young, L. J. V. Vliet, and M. V. Ginkel, "Recursive Gabor filtering," *IEEE Trans. Signal Process.*, vol. 50, no. 11, pp. 2798–2805, Nov. 2002.

[22] A. Bernardino and J. Santos-Victor, "Fast IIR isotropic 2-D complex Gabor filters with boundary initialization," *IEEE Trans. Image Process.*, vol. 15, no. 11, pp. 3338–3348, Nov. 2006.

[23] L. J. van Vliet, I. T. Young, and P. W. Verbeek, "Recursive Gaussian derivative filters," in *Proc. 14th Int. Conf. Pattern Recognit. (ICPR)*, Brisbane, QLD, Australia, Aug. 1998, pp. 509–514.

[24] X. Wang and B. E. Shi, "GPU implemention of fast Gabor filters," in *Proc. Int. Symp. Circuits Syst. (ISCAS)*, vol. 2. Paris, France, May/Jun. 2010, pp. 373–376.

[25] W.-M. Pang, K.-S. Choi, and J. Qin, "Fast Gabor texture feature extraction with separable filters using GPU," *J. Real-Time Image Process.*, vol. 12, no. 1, pp. 5–13, 2016.

[26] A. K. Gangwar and A. Joshi, "Local Gabor rank pattern (LGRP): A novel descriptor for face representation and recognition," in *Proc. IEEE Int. Workshop Inf. Forensics Secur. (WIFS)*, Rome, Italy, Nov. 2015, pp. 1–6.

[27] E. Jacobsen and R. Lyons, "The sliding DFT," *IEEE Signal Process. Mag.*, vol. 20, no. 2, pp. 74–80, Mar. 2003.

[28] E. Jacobsen and R. Lyons, "An update to the sliding DFT," *IEEE Signal Process. Mag.*, vol. 21, no. 1, pp. 110–111, Jan. 2004.

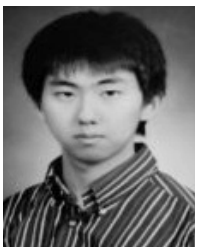
[29] C. S. Park, "2D discrete fourier transform on sliding windows," *IEEE Trans. Image Process.*, vol. 24, no. 3, pp. 901–907, Mar. 2015.

[30] I. T. Young and L. J. van Vliet, "Recursive implementation of the Gaussian filter," *Signal Process.*, vol. 44, no. 2, pp. 139–151, 1995.

[31] Univ. Southern California and I. P. Institute. *The USC-SIPI Image Database*. Accessed: Jan. 2017. [Online]. Available: http://sipi.usc.edu/services/database

[32] D. J. Field, "Relations between the statistics of natural images and the response properties of cortical cells," *J. Opt. Soc. Amer. A, Opt. Image Sci.*, vol. 4, no. 12, pp. 2379–2394, 1987.

Jaeyoon Kim received the B.S. degree from the Department of Computer Science and Engineering, Chungnam National University, Daejeon, South Korea, in 2017. He is currently pursuing the M.S. degree with the Department of Computer Science, KAIST, Daejeon, South Korea.



Suhyuk Um received the B.S. degree from the the Department of Computer Science and Engineering, Chungnam National University, Daejeon, South Korea, in 2017.

Dongbo Min (M'09–SM'15) received the B.S., M.S., and Ph.D. degrees from the School of Electrical and Electronic Engineering, Yonsei University, Seoul, South Korea, in 2003, 2005, and 2009, respectively. He was a Post-Doctoral Researcher with the Mitsubishi Electric Research Laboratories, Cambridge, MA, USA, from 2009 to 2010. From 2010 to 2015, he was with the Advanced Digital Sciences Center, Singapore. From 2015 to 2018, he was an Assistant Professor with the Department of Computer Science and Engineering, Chungnam National University, Daejeon, South Korea. Since 2018, he has been an Assistant Professor with the Department of Computer Science and Engineering, Ewha Womans University, Seoul, South Korea. His current research interests include computer vision, 2D/3D video processing, computational photography, augmented reality, and continuous/discrete optimization.