Efficient Edge Detection Using Simplified Gabor Wavelets

Wei Jiang, Kin-Man Lam, and Ting-Zhi Shen

***Abstract*—Gabor wavelets (GWs) have been commonly used for extracting local features for various applications, such as recognition, tracking, and edge detection. However, extracting the Gabor features is computationally intensive, so the features may be impractical for real-time applications. In this paper, we propose a set of simplified version of GWs (SGWs) and an efficient algorithm for extracting the features for edge detection. Experimental results show that our SGW-based edge-detection algorithm can achieve a similar performance level to that using GWs, while the runtime required for feature extraction using SGWs is faster than that with GWs with the use of the fast Fourier transform. When compared to the Canny and other conventional edge-detection methods, our proposed method can achieve a better performance in the terms of detection accuracy and computational complexity.**

***Index Terms*—Edge detection, Gabor wavelets (GWs), simplified GWs (SGWs).**

NOMENCLATURE

EE	Edge pixel in the image is detected correctly as an edge pixel.
ENE	Edge pixel is detected wrongly as a nonedge pixel.
NEE	Nonedge pixel is detected wrongly as an edge pixel.
NENE	Nonedge pixel is detected correctly as a nonedge pixel.

I. INTRODUCTION

RESEARCH in automatic edge detection has been active because of this topic's wide range of applications in image processing, such as automated inspection of machine assemblies, diagnosis in medical imaging, and topographical recognition. However, edge detection is a very difficult task. When viewing an image, humans can easily determine the boundaries within that image without needing to do so consciously. However, no single edge-detection algorithm, at present, has been devised which will successfully determine every different type of edge. Brannock and Weeks [1] have proposed an edge-detection method based on the discrete wavelet transform (DWT), which combines DWT with other methods to achieve an optimal solution to the edge-detection problem. In [2], a multiscale wavelet edge-detection algorithm was also proposed for lip segmentation.

The human visual system can be viewed as being composed of a filter bank. The responses of the respective filters can be modeled using Gabor functions of different frequencies and orientations. The Gabor features have also been found to be particularly effective for texture representation and discrimination [3], [4] and have been successfully applied to object detection [5], face recognition [6], [7], and tracking [8]–[10]. In the spatial domain, a 2-D Gabor filter is a Gaussian kernel function modulated by a sinusoidal plane wave, as follows:

$$G(x,y) = \exp\left[-\frac{x^2+y^2}{2\sigma^2}\right]\exp\left[j\omega(x\cos\theta + y\sin\theta)\right] \quad (1)$$

where σ is the standard deviation of the Gaussian function in the x- and y-directions and ω denotes the spatial frequency. With an input image $I(x,y)$, the output of the Gabor filter $\phi(x,y)$ is given as follows:

$$\phi(x,y) = G(x,y) \otimes I(x,y) \quad (2)$$

where $\otimes$ denotes the 2-D convolution operation.

The Gabor wavelets (GWs) respond strongly to edges if the edge direction is perpendicular to the wave vector $(\omega\cos\theta, \omega\sin\theta)$. When hitting an edge, the real and imaginary parts of $\phi(x,y)$ oscillate with the characteristic frequency instead of providing a smooth peak. Nevertheless, the magnitude of the response can effectively provide a measure of the image's local properties [11].

Pellegrino *et al.* [12] have proposed that the imaginary part of a Gabor filter is an efficient and robust means for edge detection. The imaginary part of a GW is

$$S(x,y) = \exp\left[-\frac{x^2+y^2}{2\sigma^2}\right]\sin\left[\omega(x\cos\theta + y\sin\theta)\right]. \quad (3)$$

To detect the edges in an image, which have different directions, θ is set to have eight different orientations, i.e., $\theta = k\pi/8$, for $k = 0, \ldots, 7$, with $\omega \cdot \sigma \leq 1$ (see the Appendix). However, extracting Gabor features is computationally intensive, so the

Manuscript received May 14, 2008; revised September 12, 2008 and November 28, 2008. First published March 24, 2009; current version published July 17, 2009. This work was supported by an internal grant from the Centre for Signal Processing, The Hong Kong Polytechnic University, Hong Kong, under Project 1-BB88. This paper was recommended by Associate Editor D. Goldgof.

W. Jiang is with the Department of Electronic Engineering, School of Information Science and Technology, Beijing Institute of Technology, Beijing 100081, China, and also with the Centre for Signal Processing, Department of Electronic and Information Engineering, The Hong Kong Polytechnic University, Kowloon, Hong Kong.

K.-M. Lam is with the Centre for Signal Processing, Department of Electronic and Information Engineering, The Hong Kong Polytechnic University, Kowloon, Hong Kong.

T.-Z. Shen is with the Department of Electronic Engineering, School of Information Science and Technology, Beijing Institute of Technology, Beijing 100081, China.

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/TSMCB.2008.2011646

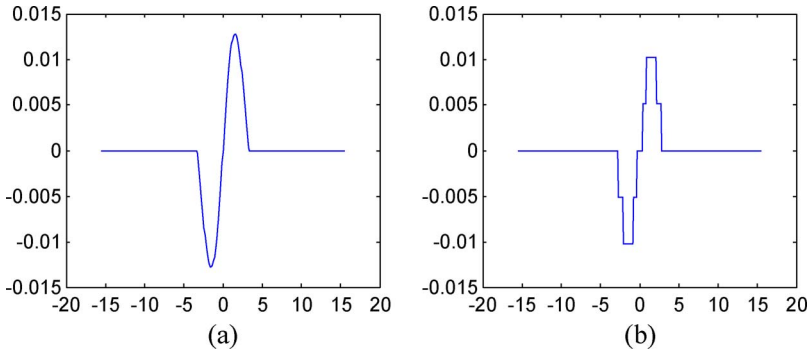


Fig. 1. (a) Imaginary part of 1-D Gabor function. (b) Corresponding quantized function.

features might be impractical for real-time applications. Therefore, in this paper, we propose a simplified version of the GWs (SGW) used for edge detection. The SGWs are in the form of simple patterns for different scales and orientations, which can be computed efficiently. In addition, these SGW features can be computed for each pixel position without requiring the use of fast Fourier transform (FFT). In the next section, we will describe our SGW-based edge-detection algorithm and, then, outline the efficient implementation of our algorithm. We also compare our proposed edge-detection algorithm with the detection based on GWs, the Sobel operators, the LoG edge detection, and the Canny edge detection.

II. SGWs for Edge Detection

In this section, we will describe the structure of our proposed SGW used for edge detection. This includes the structure of the imaginary part of an SGW, the number of quantization levels used, and the determination of the respective quantization values. SGWs have also been proposed for face recognition [13], but the parameters and the structure of the SGW for edge detection are different from those used for face recognition.

A. Shape of the Imaginary Part of an SGW

To simplify our discussion, a 1-D GW is first considered, whose equation is shown as follows:

$$S(x) = \frac{1}{2\pi\sigma^2} \exp\left[-\frac{x^2}{2\sigma^2}\right] \sin(\omega x). \tag{4}$$

Fig. 1(a) shows this imaginary part of the GW, whose values are continuous. To simplify the GW, its values are quantized to a certain number of levels. Fig. 1(b) shows a quantized SGW with two quantization levels for the positive values and two quantization levels for the negative values. Including the zero level, the total number of quantization levels used is five. The same number of quantization levels is used for the positive and the negative values of the Gabor function because it is antisymmetrical.

For 2-D cases, Fig. 2(a) shows the imaginary part of a 2-D GW, with the gray-level intensities representing the magnitudes of the Gabor function. The contours of the quantized Gabor function $S(x, y)$ are shown in Fig. 2(b).

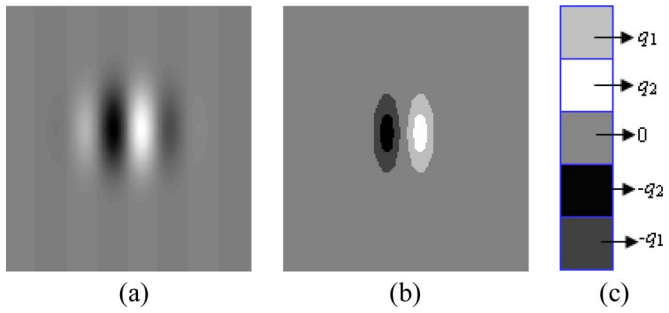


Fig. 2. (a) Imaginary part of 2-D Gabor function. (b) Corresponding contours of the quantized function. (c) Gray levels used to represent the quantized values of the five quantization levels in (b).

B. Number of Quantization Levels

The number of gray levels in an SGW depends on the number of quantization levels used to quantize the GW. If more quantization levels are employed, the SGW will be more similar to the original GW but more computation will then be involved for feature extraction. In other words, there is a tradeoff between computation and approximation accuracy. In the following sections, we will consider five quantization levels in our discussion. Then, in Section V, we will present the performance of the proposed SGW-based edge-detection algorithm with different numbers of quantization levels.

C. Determination of Quantization Levels

The determination of the quantization levels for an SGW is the same as that in [13]. One of the quantization levels of the SGW is set to zero. As the imaginary part of a Gabor function is antisymmetrical, the number of quantization levels for the positive and negative values are equal and are denoted as n_l. Then, the total number of quantization levels is $2n_l + 1$. Suppose that the largest magnitude of the GW is A, the corresponding quantization levels for positive levels $q_+(k)$ and negative levels $q_-(k)$ are as follows:

$$q_+(k) = \frac{A}{2n_l + 1} \cdot 2k \quad q_-(k) = -\frac{A}{2n_l + 1} \cdot 2k \tag{5}$$

where $k = 1, \ldots, n_l$. These SGWs are then convolved with an image to extract the SGW features at different center frequencies and orientations to form a simplified Gabor jet.

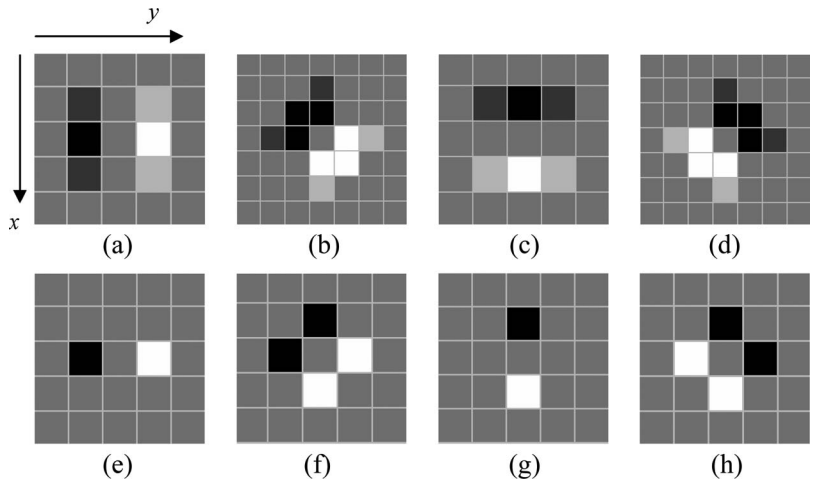


Fig. 3. Masks of the SGWs with (a) $\omega = 0.3\ \pi$ and $\theta = 0$, (b) $\omega = 0.3\ \pi$ and $\theta = \pi/4$, (c) $\omega = 0.3\ \pi$ and $\theta = \pi/2$, (d) $\omega = 0.3\ \pi$ and $\theta = 3\pi/4$, (e) $\omega = 0.5\ \pi$ and $\theta = 0$, (f) $\omega = 0.5\ \pi$ and $\theta = \pi/4$, (g) $\omega = 0.5\ \pi$ and $\theta = \pi/2$, and (h) $\omega = 0.5\ \pi$ and $\theta = 3\pi/4$.

D. Determination of the Parameters

In this section, we will determine the values of important parameters for the GWs or SGWs for edge detection, which are the values of ω, σ, and θ. To detect the edges of an image, which have different directions, we need to set θ to have different orientations. Hence, for simplicity's sake, the number of orientations used in our algorithm is four, i.e., $\theta_k = k\pi/4$ for $k = 0, \ldots, 3$. According to the condition derived in the Appendix for using the odd Gabor function for edge detection, we have $\omega \cdot \sigma \leq 1$. As edges are very localized feature of an image, the value of ω should be small when compared to that for face recognition. Therefore, for detecting edges of different scales, we set $\omega = 0.3\ \pi$ and 0.5π for two different scales. In Section V, we will evaluate the performance of our algorithm with the use of several different scales.

III. Use of SGWs for Edge Detection

In the previous section, we have presented the structure of SGWs and the parameters to be used. In this section, we will first describe feature extraction using GWs and SGWs. The patterns of the SGWs at different scales and orientations will be illustrated, and a fast algorithm for using SGWs for edge detection will also be presented.

A. Feature Extraction Using GWs

By selecting different center frequencies and orientations, we can generate a family of GW kernels using (3), which is then used for feature extraction from images. Given a gray-level image $I(x, y)$, the GW features are extracted by convolving $I(x, y)$ with each of the GWs, as in (2). The convolution can be computed efficiently using FFT, then point-by-point multiplications, and finally the inverse FFT (IFFT). By concatenating the convolution outputs, we can obtain a GW feature vector $\phi_{\omega,\theta}$ of dimension $N_w \cdot N_H$ as follows:

$$\phi_{\omega,\theta} = [\phi_{\omega,\theta}(0,0), \phi_{\omega,\theta}(0,1), \ldots, \phi_{\omega,\theta}(0, N_H - 1), \phi_{\omega,\theta}(1,0), \ldots, \phi_{\omega,\theta}(N_W - 1, N_H - 1)]^{\mathrm{T}} \quad (6)$$

where T represents the transpose operation and N_W and N_H are the width and height of the image, respectively. Although FFT is applied to reduce the computational complexity, it is still very computationally intensive. For n_{s} different scales and n_{o} different orientations, the number of GWs involved is $n_{\mathrm{s}} \cdot n_{\mathrm{o}}$. In addition, the size of the image concerned must be a power of two so as to be able to use FFT in the computation.

B. Edge Detection Using SGWs

In our algorithm, we employ SGWs of two different scales and four different orientations. The convolution of an SGW of scale ω and orientation θ with the image $I(x, y)$ generates the SGW features and is denoted as $\phi'_{\omega,\theta}(x, y)$. The resulting SGW feature $\phi''_{\omega,\theta}(x, y)$ at a pixel position (x_c, y_c) is equal to the absolute maximum of the eight $\phi'_{\omega,\theta}(x_c, y_c)$, i.e.,

$$\phi''_{\omega,\theta}(x_c, y_c) = \max\left\{\phi'_{\omega_i,\theta_j}(x_c, y_c),\ i = 0, 1 \text{ and } j = 0, \ldots, 3\right\} \quad (7)$$

where $\omega_0 = 0.3\ \pi$, $\omega_1 = 0.5\ \pi$, and $\theta_j = \pi j/4$, for $j = 0, \ldots, 3$. The SGW feature $\phi'_{\omega,\theta}(x, y)$ is computed by convolving the image $I(x, y)$ with the SGW whose patterns are dependent on the scale ω and the orientation θ. As edges are very localized in an image, so the window size of the patterns is either 3×3 or 5×5. The SGWs are formed using two levels of quantization for the positive and the negative magnitudes of the GWs. As shown in Fig. 1(b), these two quantization levels are denoted as q_1 and q_2 with $q_2 > q_1$ for positive magnitudes, and the corresponding quantization levels for the negative magnitude are $-q_1$ and $-q_2$, respectively. In our algorithm, two different scales and four different orientations are adopted. The corresponding patterns of the eight SGWs are shown in Fig. 3. Suppose that an SGW pattern is moved to the pixel position (x_c, y_c), the computation of the output $\phi'_{\omega,\theta}(x_c, y_c)$ at this position can be computed as follows.

1) $\omega = 0.3\ \pi$ and $\theta = 0$ [Fig. 3(a)]

$$\begin{aligned}\phi'_{0,0}(x_c, y_c) = q_1 &\left(I(x_c - 1, y_c + 1) + I(x_c + 1, y_c + 1)\right. \\ &\left. - I(x_c - 1, y_c - 1) - I(x_c + 1, y_c - 1)\right) \\ &+ q_2\left((I(x_c, y_c + 1) - I(x_c, y_c - 1)\right). \end{aligned} \quad (8)$$

2) $\omega = 0.3\,\pi$ and $\theta = \pi/4$ [Fig. 3(b)]

$$\begin{aligned}\phi'_{0,1}(x_c, y_c) = {} & q_1\left(I(x_c, y_c + 2) + I(x_c + 2, y_c)\right. \\ & \left. - I(x_c, y_c - 2) - I(x_c - 2, y_c)\right) \\ & + q_2\left(I(x_c, y_c + 1) + I(x_c + 1, y_c)\right. \\ & \left. + I(x_c + 1, y_c + 1) - I(x_c, y_c - 1)\right. \\ & \left. + I(x_c - 1, y_c) - I(x_c - 1, y_c - 1)\right).\end{aligned} \tag{9}$$

3) $\omega = 0.3\,\pi$ and $\theta = \pi/2$ [Fig. 3(c)]

$$\begin{aligned}\phi'_{0,2}(x_c, y_c) = {} & q_1\left(I(x_c + 1, y_c - 1) + I(x_c + 1, y_c + 1)\right. \\ & \left. - I(x_c - 1, y_c - 1) - I(x_c - 1, y_c + 1)\right) \\ & + q_2\left(I(x_c + 1, y_c) - I(x_c - 1, y_c)\right).\end{aligned} \tag{10}$$

4) $\omega = 0.3\,\pi$ and $\theta = 3\pi/4$ [Fig. 3(d)]

$$\begin{aligned}\phi'_{0,3}(x_c, y_c) = {} & q_1\left(I(x_c, y_c - 2) + I(x_c + 2, y_c)\right. \\ & \left. - I(x_c - 2, y_c) - I(x_c, y_c + 2)\right) \\ & + q_2\left(I(x_c, y_c - 1) + I(x_c + 1, y_c)\right. \\ & \left. + I(x_c + 1, y_c - 1) - I(x_c - 1, y_c)\right. \\ & \left. - I(x_c - 1, y_c + 1) - I(x_c, y_c + 1)\right).\end{aligned} \tag{11}$$

5) $\omega = 0.5\,\pi$ and $\theta = 0$ [Fig. 3(e)]

$$\phi'_{1,0}(x_c, y_c) = q_2\left(I(x_c, y_c + 1) - I(x_c, y_c - 1)\right). \tag{12}$$

6) $\omega = 0.5\,\pi$ and q [Fig. 3(f)]

$$\begin{aligned}\phi'_{1,1}(x_c, y_c) = {} & q_2\left(I(x_c, y_c + 1) + I(x_c + 1, y_c)\right. \\ & \left. - I(x_c, y_c - 1) - I(x_c - 1, y_c)\right).\end{aligned} \tag{13}$$

7) $\omega = 0.5\,\pi$ and $\theta = \pi/2$ [Fig. 3(g)]

$$\phi'_{1,2}(x_c, y_c) = q_2\left(I(x_c + 1, y_c) - I(x_c - 1, y_c)\right). \tag{14}$$

8) $\omega = 0.5\,\pi$ and $\theta = 3\pi/4$ [Fig. 3(h)]

$$\begin{aligned}\phi'_{1,3}(x_c, y_c) = {} & q_2\left(I(x_c, y_c - 1) + I(x_c + 1, y_c + 1)\right. \\ & \left. - I(x_c, y_c + 1) - I(x_c - 1, y_c)\right).\end{aligned} \tag{15}$$

As shown from (8)–(15), the computation of a $\phi'_{\omega,\theta}(x_c, y_c)$ requires no more than 2 multiplications and 22 additions. If the operations required for generating the pixel coordinates are not considered, the number of additions required is ten only. These $\phi'_{\omega,\theta}(x_c, y_c)$ identify the edges of two scales and four orientations in images. In the next section, we will present an efficient way to determine edge pixels using SGW features.

C. Efficient Edge Detection

As described in Section III-B, the computation of the SGW features is computationally simple, and the feature can be computed for each pixel position individually. Hence, the computation required for SGW features is much lower than that required for GW features. However, four different orientations are adopted for each SGW. Usually, the number of edge pixels is much smaller than that of nonedge pixels. If only two orientations of a single scale are used to determine whether a pixel is a potential edge or not, a lot of computation can then be saved. In our algorithm, we first select the two SGWs with orientations $\theta = 0$ and $\pi/2$ and with scale $\omega = 0.3\,\pi$, i.e., $\phi'_{0,0}(x, y)$ and $\phi'_{0,2}(x, y)$ are computed. If a pixel position is an edge, either $\phi'_{0,0}(x, y)$ or $\phi'_{0,2}(x, y)$ should be larger than a certain threshold T_1. In other words, if both $|\phi'_{0,0}(x, y)| < T_1$ and $|\phi'_{0,2}(x, y)| < T_1$ at a pixel position, then it is not an edge point. Hence, other SGW features $\phi'_{\omega,\theta}(x, y)$ need not be computed, and the overall computation required can be greatly reduced. If the pixel position under consideration is a strong edge, the sum of $|\phi'_{0,0}(x, y)| + |\phi'_{0,2}(x, y)|$ will be larger than another threshold T_2, i.e., $|\phi'_{0,0}(x, y)| + |\phi'_{0,2}(x, y)| > T_2$. In this case, the point has been identified as an edge, so no further computation will be needed. Otherwise, the other six $\phi'_{\omega,\theta}(x, y)$ will be computed to determine whether the point is an edge or not by the use of thresholding. The thresholds T_1 and T_2 are important parameters for the performance of the edge-detection algorithm; these are determined by using the minimax threshold method [14]. In our algorithm, we set $T_2 = 2T_1$, and T_1 can be obtained using the method presented in [15]. In the next section, we will analyze the computational requirements for using GWs and SGWs for edge detection, as well as the Canny algorithm. Then, the performances of different edge-detection algorithms will be evaluated and compared.

IV. Computation Analysis of Edge-Detection Algorithm

In this section, we will analyze and compare the computational complexities of three different edge-detection algorithms: the Canny algorithm and that based on GWs and on SGWs. Within our context, the computation of an algorithm refers to the number of real additions and real multiplications required for edge detection. The number of additions does not count the operations required for generating the pixel coordinates in our analysis. As FFT is employed for the case of GW-based edge detection, we assume that the image size is a power of two. For the cases of the SGW-based and the Canny algorithms, the image may be any size. The SGW features at any pixel position can be computed individually and efficiently.

A. Canny Algorithm

The Canny edge-detection algorithm [16] is known to many as an optimal edge detector. However, the computational cost required is not small. There are five steps in the Canny edge-detection algorithm; only the first three of these involve additions and multiplications. The first step is to filter out any noise in an image before detecting edges by using a Gaussian mask with a size of 5×5. For an image of size $N \times N$, $24N^2$ real additions and $6N^2$ real multiplications are required. The second step is to determine the edge strength by taking the gradient

TABLE I
COMPUTATIONAL COMPLEXITIES OF FEATURE EXTRACTION USING NR-SGW AND R-SGW WITH DIFFERENT SCALES AND ORIENTATIONS

		No. of additions	No. of multiplications
$\omega = 0.3\pi$	$\theta = 0$ or $\pi/2$	$5N^2$	$4N^2$
	$\theta = \pi/4$ or $3\pi/4$	$9N^2$	$4N^2$
$\omega = 0.5\pi$	$\theta = 0$ or $\pi/2$	N^2	$4N^2$
	$\theta = \pi/4$ or $3\pi/4$	$3N^2$	$4N^2$

of the image in the x- and y-directions, which needs $10N^2$ real additions and $8N^2$ real multiplications. The third step is to compute the edge direction, where N^2 real multiplications are required.

B. GW-Based Edge Detection

Given an image $f(x, y)$ of size $N \times N$ and the imaginary part of a GW $g(x, y)$ with an arbitrary scale and orientation, the GW features can be extracted using convolution, i.e., $f(x, y) \otimes g(x, y)$. The convolution can be implemented using FFT, then point-by-point multiplications, and finally IFFT. In our analysis, we assume that the FFTs of all the GWs have been precomputed.

Performing FFT on an $N \times N$ image requires $N^2 \log_2 N^2$ complex additions and $N^2 \log_2 N^2/2$ complex multiplications. The IFFT requires the same amount of computation as the FFT. The point-by-point multiplications involve N^2 complex multiplications. Therefore, feature extraction based on a GW requires a total of $2N^2 \log_2 N^2$ complex additions and $N^2 \log_2 N^2 + N^2$ complex multiplications. Performing one complex addition requires two real additions, while one complex multiplication requires two real additions and four real multiplications. If the number of scales and orientations used is n_s and n_o, respectively, then the computation required is approximately $n_\mathrm{s} \cdot n_\mathrm{o} \cdot (6N^2 \log_2 N^2 + 2N^2)$ real additions and $n_\mathrm{s} \cdot n_\mathrm{o} \cdot (4N^2 \log_2 N^2 + 4N^2)$ real multiplications.

C. SGW-Based Edge Detection

As described in Section III-B, efficient ways are available for extracting the SGW features using SGWs at four different orientations. The SGW features of different scales and orientations can be computed efficiently using (8)–(15). It can be seen that the computation required for the orientations 0 or $\pi/2$ [i.e., nonrotated SGWs (NR-SGWs)] is lower than that required for the orientations $\pi/4$ or $3\pi/4$ [i.e., rotated SGWs (R-SGWs)]. This is because an NR-SGW involves a smaller number of pixels in extracting the SGW feature as compared to the corresponding R-SGW. For extracting SGW features from an image of size $N \times N$, the respective computations required when using an NR-SGW and an R-SGW with the two different scales are tabulated in Table I.

Table II illustrates the computational complexities of the three edge-detection algorithms: Canny, GW, and SGW. The number of scales and orientations used for GW and SGW are two and four, respectively. We can see that our SGW-based approach requires the smallest amount of computation, while the GW-based method is the most computational. For the SGW-based approach, the numbers shown in the table are the maximum number of operations needed. The computation level depends on the complexity of the image concerned. In Section V, we will also compare the runtimes of the respective approaches.

TABLE II
COMPUTATIONAL COMPLEXITIES OF THE CANNY, GW-BASED, AND SGW-BASED EDGE-DETECTION ALGORITHMS

Algorithms	No. of additions	No. of multiplications
Canny	$40N^2$	$17N^2$
GW	$48N^2\log_2N^2$+$16N^2$	$32N^2\log_2N^2$+$32N^2$
SGW	$18N^2$	$16N^2$

V. EXPERIMENTAL RESULTS

In this section, we will first evaluate the performance of the proposed SGW-based approach with different numbers of quantization levels, as well as different scales. Then, the relative performances with the use of the GW features, the SGW features, and the SGW-based fast algorithm will be compared. Finally, we will compare the performances of the Canny, the GW-based, and the SGW-based edge-detection algorithms in terms of detection accuracy and runtime.

A. Relative Performances of SGWs With Different Quantization Levels for Edge Detection

To evaluate the effect of the number of quantization levels on edge detection when using SGWs, three different quantization levels are considered. As the absolute magnitude of the positive and negative parts of an imaginary Gabor function is the same, so one, two, and three levels are selected for both the positive and the negative parts. Including the level of zero magnitude, the quantization levels used in this experiment are three, five, and seven. In this experiment, the images "Lena" and "fish," as shown in Fig. 4(a), are used.

Fig. 4(b)–(e) shows the SGW features based on scale $\omega = 0.3\ \pi$, three quantization levels, and four different orientations

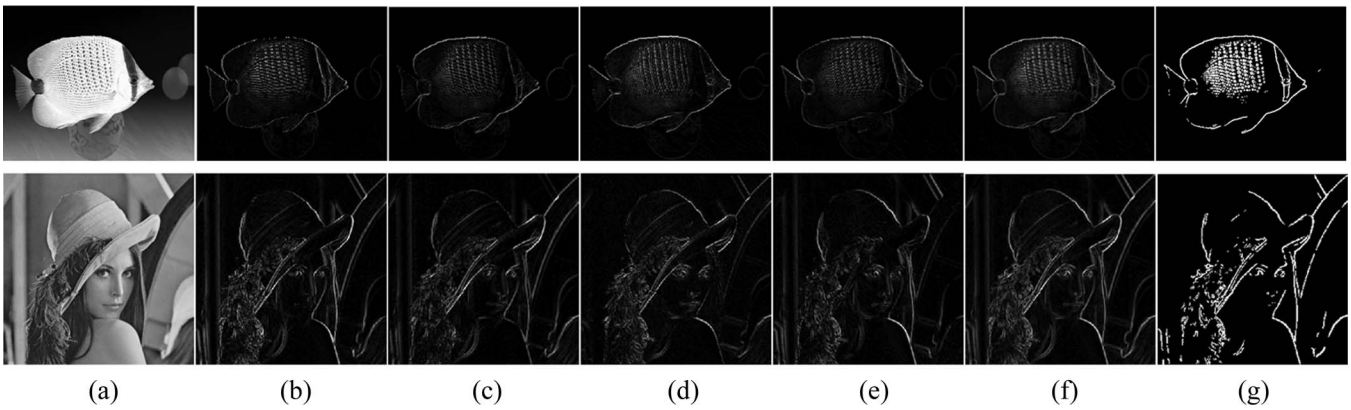


Fig. 4. SGW features and edge-detection result for $\omega = 0.3\ \pi$ and three quantization levels. (a) Original images. (b) $\theta = 0$. (c) $\theta = \pi/4$. (d) $\theta = \pi/2$. (e) $\theta = 3\pi/4$. (f) Maximum of the SGW features. (g) Result after thresholding.

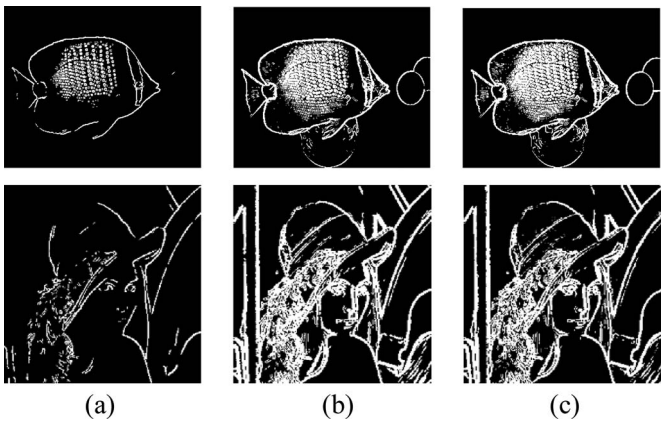


Fig. 5. Edge-detection results based on SGWs with $\omega = 0.3\ \pi$ and three different quantization levels. (a) Three levels. (b) Five levels. (c) Seven levels.

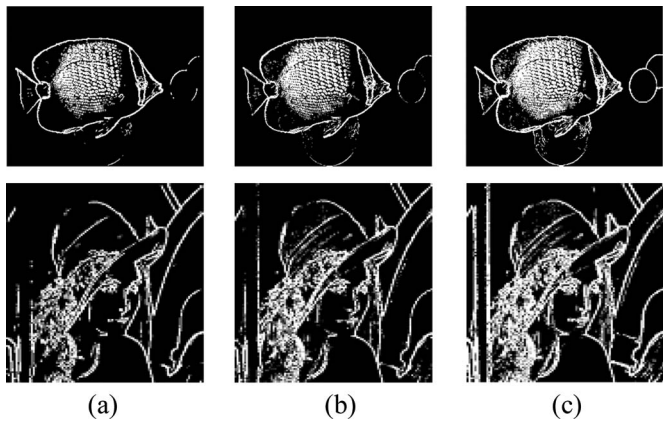


Fig. 6. Edge-detection results based on SGWs with $\omega = 0.5\ \pi$ and three different quantization levels. (a) Three levels. (b) Five levels. (c) Seven levels.

applied to the two images. Fig. 4(f) is generated by choosing the maximum from the four SGW features. Fig. 4(g) is the edge-detection result after thresholding the result shown in Fig. 4(f). Figs. 5 and 6 show the detection results when the number of quantization levels used are three, five, and seven for $\omega = 0.3\ \pi$ and $\omega = 0.5\ \pi$, respectively. From Figs. 5 and 6, we can see that the performances when using SGWs of five and seven quantization levels are better than that when using three quantization levels, while the performances for five and seven quantization levels are very similar. Using a higher number of quantization levels can approximate the Gabor functions more accurately but will require more computation. Hence, five quantization levels are chosen in our algorithm.

B. Relative Performances of SGWs With Different Scales

In the previous section, we have found that SGWs with five quantization levels can give the most promising performance in terms of accuracy and computation. Therefore, we will evaluate the SGW-based edge detection with five different quantization levels and with different scales: $\omega = 0.125\ \pi$, $\omega = 0.3\ \pi$, $\omega = 0.5\ \pi$, and $\omega = 0.65\ \pi$.

Fig. 7 shows the edge images of the images "Lena" and "fish" based on SGWs of the four different scales. The scale ω should be set at a much smaller value than that for face recognition, because the edges of an image are very localized features. However, if ω is set at a much smaller value, σ in (4) will become very large. This means that more pixels must be involved in computing the SGW features $\phi'_{\omega,\theta}$ in (7), and so, the edges will be blurred. Consequently, we set $\omega = 0.3\ \pi$ and $\omega = 0.5\ \pi$ in our algorithm; these have between two and ten pixels involved in computing $\phi'_{\omega,\theta}$, as shown in Fig. 3.

C. Comparing the Performances of the SGW Features and the GW Features for Edge Detection

The use of SGWs for edge detection can save a lot of computation as compared to GWs, while maintaining a level of detection accuracy comparable with the GW. Fig. 8(a) and (b) shows the edge-detection results using GWs and SGWs with the same center frequencies and orientations. The number of orientations is also set at four, and the number of quantization levels used for SGWs is five. Fig. 8(c) shows the results using SGWs with the efficient scheme. The edge images generated are then postprocessed using a thinning algorithm based on morphological operations [17] so that the edges are of one-pixel width. The corresponding results are shown in Fig. 8(d)–(f). From the different detection results, we see that the relative

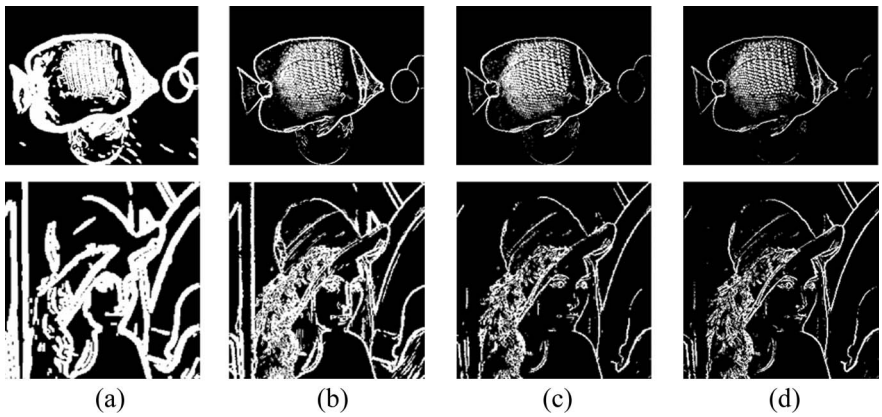


Fig. 7. SGW-based edge detection with three different scales. (a) $\omega = 0.125\,\pi$. (b) $\omega = 0.3\,\pi$. (c) $\omega = 0.5\,\pi$. (d) $\omega = 0.65\,\pi$.

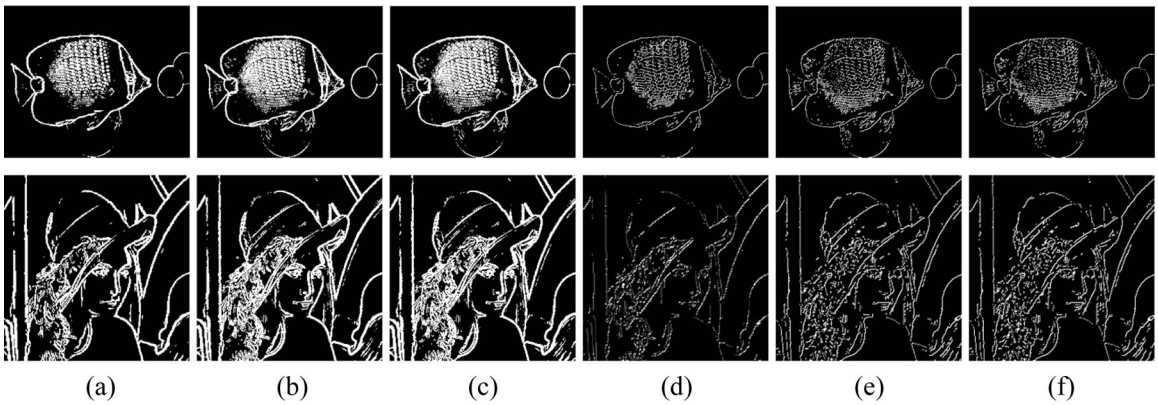


Fig. 8. Edge-detection results based on (a) GWs, (b) SGWs, and (c) SGWs with the efficient scheme, (d) GWs with thinning, (e) SGWs with thinning, and (f) efficient scheme with thinning.

performances of the SGWs and the GWs with the same center frequencies and orientations are very similar. In some cases, the detection based on SGWs outperforms the GWs. The center frequency of a GW function and its SGW version should be very similar. An SGW is a quantized version of its GW; their rates of variation should be maintained. Hence, in the frequency domain, the center frequencies of the SGW and the GW should be very close, while the shape of their spectra will differ. In other words, the features extracted using a GW and its corresponding SGW should be similar. In addition, we cannot find any obvious difference between the detection results based on SGWs with and without using the efficient scheme.

D. Runtimes for Different Edge-Detection Algorithms

In this section, we compare the runtimes required by the SGW-based, the GW-based, and the Canny edge detectors. The image "Lena," with a size of 256×256, was used. The respective algorithms were executed 10 000 times, and the respective average runtimes are computed. Table III tabulates the respective runtimes for the three different algorithms. The runtime required by our algorithm with using the efficient scheme is smaller than that required by our algorithm without using the efficient scheme. When three-level-quantized SGWs with the efficient scheme are employed, the speedup rates are 358 and 1.93 as compared to the GW-based and the Canny edge detector. When more quantization levels are considered, the speedup rate of the SGW-based algorithm will decrease.

Based on the earlier experiments, we can conclude that the performance of our SGW-based algorithm in terms of detection accuracy is comparable to that of the GW-based algorithm, while the amount of computation required is significantly smaller. GWs can extract features which are discriminative and useful for many applications, but they are impractical for real-time applications due to their high complexity in feature extraction. Consequently, SGWs can be propelled to replace GWs for real-time applications and processing.

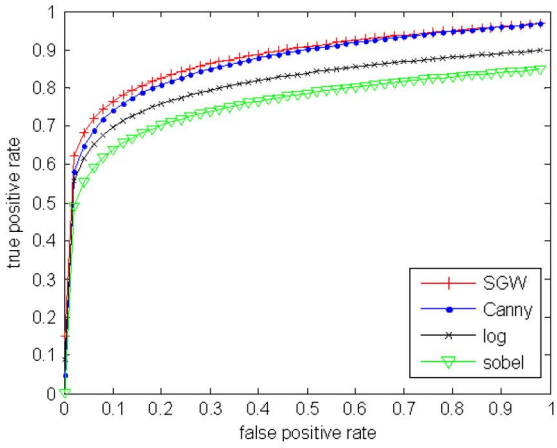
E. Comparing the Performance of the SGW-Based Algorithm to Other Edge-Detection Algorithms

In this section, we compare the performance of our SGWs-based edge-detection algorithm with some conventional edge-detection algorithms, including the Sobel edge detector, the LoG edge detection, and the Canny edge detector. The SGWs employ two scales $\omega = \{0.3\pi, 0.5\pi\}$, four orientations $\theta = \{0, \pi/4, \pi/2, 3\pi/4\}$, and five quantization levels. In our experiments, the parameters used in the Canny edge detector are the two thresholds—the low threshold and the high threshold—and

TABLE III
AVERAGE RUNTIMES OF DIFFERENT EDGE-DETECTION ALGORITHMS: GW, SGW, AND CANNY. BOTH THE GWS AND THE SGWS (WITH AND WITHOUT USING THE EFFICIENT SCHEME) EMPLOY TWO SCALES AND FOUR ORIENTATIONS, AND THE SGWS ALSO USE DIFFERENT NUMBERS OF QUANTIZATION LEVELS

	3 quantization levels	5 quantization levels	7 quantization levels
SGW (without using the efficient scheme)	15.21ms	20.24ms	27.87ms
SGW (with using the efficient scheme)	12.35ms	16.87ms	23.33ms
GW	4,422ms		
Speed-up rate of SGW (with the efficient scheme)	358	262	189
Canny	23.84ms		
Speed-up rate of SGW (with the efficient scheme)	1.93	1.41	1.02

(a) (b)



(c)

Fig. 9. ROC curves. (a) Original image. (b) Corresponding ground truth. (c) ROC curves of our SGW-based edge detector: The Sobel edge detector, the LoG edge detector, and the Canny edge detector.

the standard deviation of a Gaussian filter used. The high threshold and the low threshold are set at 0.1 and 0.04, respectively, and the standard deviation is set at one in the experiments. This optimal setting is determined according to the Canny method in [16].

To evaluate the accuracy of the different edge-detection algorithms, we use the receiver-operating-characteristic curve (or ROC curve), as proposed in [18] and [19] to measure the edge-detection accuracy. For every pixel in an image, there are four possible edge-detection results, as shown in the Nomenclature.

Hence, the ROC curve can be used to reflect the relationship between the true-positive (TP) and the false-positive (FP) rates of an edge-detection algorithm, where TP and FP are defined as follows:

$$\mathrm{TP} = \frac{n_{\mathrm{EE}}}{n_{\mathrm{EE}} + n_{\mathrm{ENE}}} \quad \mathrm{FP} = \frac{n_{\mathrm{NEE}}}{n_{\mathrm{NEE}} + n_{\mathrm{NENE}}} \tag{16}$$

where n_{EE}, n_{ENE}, n_{NEE}, and n_{NENE} are the number of pixels for the four results, i.e., EE, ENE, NEE, and NENE, respectively.

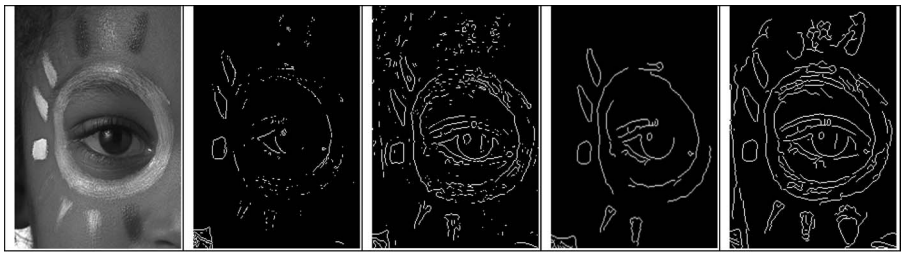
Fig. 10. Edge images detected using different algorithms. (a) Original images. (b) Sobel edge detector. (c) LoG edge detector. (d) Canny edge detector. (e) SGW-based edge detector.

Fig. 9(a) and (b) shows one of the original images and the corresponding ground truth from the data set in [18] used in our experiment. Fig. 9(c) shows the ROC curves of our SGW-based edge detector, the Sobel edge detector, the LoG edge detector, and the Canny edge detector. We can see that when the FP prediction rate is between 0 and 0.1, the TP rate of our proposed algorithm increases more sharply than that of the other three methods. In addition, when the FP rate is between 0 and 0.5, our algorithm outperforms the other three algorithms, and when the FP rate is higher than 0.5, our performance is either slightly better than, or the same as, the Canny edge detector.

Fig. 10 shows the detection results using seven other images shown in Fig. 10(a). It can be seen that these conventional algorithms cannot detect blurry edges accurately, while our method can detect the edges at different orientations to produce accurate edge-detection results. In general, the edge images generated by our method are smooth, and the edges are more continuous. In Fig. 11, we enlarge the right-eye region of the first image in Fig. 10 so that we can compare the respective algorithms visually. We can see that the edges detected using our SGW-based edge detector are smoother and continuous than the edge images detected using the other three methods.

VI. Conclusion

In this paper, we have proposed using features based on a simplified version of GWs for edge detection and have introduced the masks for the SGWs at different scales and orientations for edge detection. We have also presented a fast implementation of our algorithm by noting that the features at different orientations are closely related. Hence, SGW features of two orientations at each pixel position are usually required. Our efficient edge-detection algorithm can achieve a performance level in terms of detection accuracy similar to that based on the original GWs, but it requires a significantly smaller amount of computation. When two center frequencies and four orientations are employed, the relative performances



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

Fig. 11. Highlight of the edges detected using different algorithms. (a) Original images. (b) Sobel edge detector. (c) LoG edge detector. (d) Canny edge detector. (e) SGW-based edge detector.

of the SGWs and the GWs are very similar. A speedup rate of about 262 times can be achieved if five-level-quantized SGWs are used. With the image "Lena" of size 256 × 256, the runtimes required for edge detection based on SGWs with five quantization levels and on GWs are 16.87 ms and 4422 ms, respectively. Our algorithm is also compared to the Canny algorithm and other conventional algorithms, and ours, again, shows a superior performance.

APPENDIX

In order to determine the optimal values of the parameters ω, σ, and θ, we consider an image in the form of a 2-D step signal. The 2-D step signal is defined as follows:

$$U(x,y)=\begin{cases}1, & y\le 0\\ 0, & y>0.\end{cases} \tag{A.1}$$

Hence, the edge is parallel to the x-axis. $S(x,y)$ is convolved with $U(x,y)$ to extract the GW feature, i.e., $f(x,y)=S(x,y)\otimes U(x,y)$ is computed. To detect the edges of the step signal, the convolution output f at $(x=0)$ should have the maximum magnitude. There, the parameters ω, σ, and θ are set in such a way as to maximize f at $(x=0)$. The values of these parameters are determined as follows:

$$\begin{aligned}f(x,y)&=S(x,y)\otimes U(x,y)\\ &=\int_{-\infty}^{+\infty}\int_{-\infty}^{+\infty}U(\tau_1,\tau_2)S(x-\tau_1,y-\tau_2)d\tau_1 d\tau_2\\ &=\int_{-\infty}^{+\infty}\int_{-\infty}^{0}S(x-\tau_1,y-\tau_2)d\tau_1 d\tau_2.\end{aligned} \tag{A.2}$$

We replace $x-\tau_1$ with u and $y-\tau_2$ with v, then (A.3) is presented, shown at the bottom of this page.

The integral is too hard to compute, so we use the following Taylor series to approximate the sine and cosine functions, i.e.,

$$\sin u = u-\frac{1}{3!}u^3+\cdots+(-1)^n\frac{1}{(2n+1)}u^{2n+1}+\cdots \tag{A.4}$$

$$\cos u = 1-\frac{1}{2!}u^2+\frac{1}{4!}u^4+\cdots+(-1)^n\frac{1}{(2n)!}u^{2n}+\cdots. \tag{A.5}$$

To have an accurate approximation, we set $n=3$, then (A.6) is presented, shown at the top of the next page.

We have to determine the values of ω, σ, and θ, which will maximize f at $(x=0)$. These can be obtained by differentiating the equation $f(x=0,y)$ with respect to x and y and, then,

$$\begin{aligned}f&=\int_{-\infty}^{+\infty}\int_{-y}^{+\infty}\exp\left[-\frac{(u^2+v^2)}{2\sigma^2}\right]\sin\left[\omega(u\cos\theta+v\sin\theta)\right]dudv\\ &=\int_{-\infty}^{+\infty}\int_{-y}^{+\infty}\exp\left[-\frac{(u^2+v^2)}{2\sigma^2}\right]\left[\sin(\omega u\cos\theta)\cos(\omega v\sin\theta)+\cos(\omega u\cos\theta)\sin(\omega v\sin\theta)\right]dudv\\ &=\int_{-\infty}^{+\infty}\exp\left(-\frac{u^2}{2\sigma^2}\right)\sin(\omega u\cos\theta)du\int_{y}^{+\infty}\exp\left(-\frac{v^2}{2\sigma^2}\right)\cos(\omega v\sin\theta)dv\\ &\quad+\int_{-\infty}^{+\infty}\exp\left(-\frac{u^2}{2\sigma^2}\right)\cos(\omega u\cos\theta)du\int_{y}^{+\infty}\exp\left(-\frac{v^2}{2\sigma^2}\right)\sin(\omega v\sin\theta)dv\end{aligned} \tag{A.3}$$

$$
\begin{aligned}
f(x,y) &= S(x,y) \otimes U(x,y) \\
&= -\sqrt{2\pi}\sigma \exp\left(-\frac{\omega^2 \cos^2(\theta)\sigma^2}{2}\right) \left\{ \left[\omega \sin\theta - \frac{1}{3!}(\omega \sin\theta)^3 y^4 + \frac{1}{5!}(\omega \sin\theta)^5 y^8\right](-2\sigma^2) \exp\left(-\frac{y^2}{2\sigma^2}\right) \right. \\
&\quad - \left[-\frac{1}{3}(\omega \sin\theta)^3 y^2 + \frac{4}{5!}(\omega \sin\theta)^5 y^6\right](-2\sigma^2)^2 \exp\left(-\frac{y^2}{2\sigma^2}\right) \\
&\quad + \left[-\frac{1}{3}(\omega \sin\theta)^3 + \frac{1}{10}(\omega \sin\theta)^5 y^4\right](-2\sigma^2)^3 \exp\left(-\frac{y^2}{2\sigma^2}\right) \\
&\quad \left. - \frac{16}{5}(\omega \sin\theta)^5 y^2 \sigma^8 \exp\left(-\frac{y^2}{2\sigma^2}\right) + \frac{1}{5}(\omega \sin\theta)^5 (-2\sigma^2)^5 \exp\left(-\frac{y^2}{2\sigma^2}\right) \right\}
\end{aligned} \tag{A.6}
$$

setting them to zero. Hence, the following four equations are computed:

$$\frac{\partial f(x=0,y)}{\partial x} = 0 \tag{A.7}$$

$$\frac{\partial f(x=0,y)}{\partial y} = 0 \tag{A.8}$$

$$
\begin{aligned}
&\left(\frac{\partial^2 f(x=0,y)}{\partial x \partial y}\right)^2 - \frac{\partial^2 f(x=0,y)}{\partial x^2} \\
&\quad \bullet \frac{\partial^2 f(x=0,y)}{\partial y^2} < 0
\end{aligned} \tag{A.9}
$$

$$\frac{\partial^2 f(x=0,y)}{\partial y^2} < 0. \tag{A.10}$$

The results of these four equations are that $\theta = \pi/2$ and $\omega \cdot \theta \leq 1$. Hence, the imaginary part of a Gabor filter can achieve the best performance for edge detection when $\omega \cdot \theta \leq 1$ and θ is perpendicular to the edge.

REFERENCES

[1] E. Brannock and M. Weeks, "A synopsis of recent work in edge detection using the DWT," in *Proc. IEEE Southeastcon*, 2008, pp. 515–520.

[2] Y. P. Guan, "Automatic extraction of lips based on multi-scale wavelet edge detection," *Comput. Vis.*, vol. 2, no. 1, pp. 23–33, Mar. 2008.

[3] A. C. Bovik, M. Clark, and W. S. Geisler, "Multichannel texture analysis using localized spatial filters," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 12, no. 1, pp. 55–73, Jan. 1990.

[4] B. S. Manjunath and W. Y. Ma, "Texture features for browsing and retrieval of image data," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 18, no. 8, pp. 837–842, Aug. 1996.

[5] H. Cheng, N. Zheng, and C. Sun, "Boosted Gabor features applied to vehicle detection," in *Proc. 18th Int. Conf. Pattern Recog.*, 2006, vol. 1, pp. 662–666.

[6] M. Valstar and M. Pantic, "Fully automatic facial action unit detection and temporal analysis," in *Proc. IEEE Conf. CVPRW*, Jun. 2006, p. 149.

[7] X. Xie and K. M. Lam, "Gabor-based kernel PCA with doubly nonlinear mapping for face recognition with a single face image," *IEEE Trans. Image Process.*, vol. 15, no. 9, pp. 2481–2492, Sep. 2006.

[8] E. Painkras and C. Charoensak, "A framework for the design and implementation of a dynamic face tracking system," in *Proc. IEEE TENCON*, Nov. 2005, pp. 1–6.

[9] S. Dubuisson, "An adaptive clustering for multiple object tracking in sequences in and beyond the visible spectrum," in *Proc. IEEE Conf. CVPRW*, Jun. 2006, p. 142.

[10] D. Tao, X. Li, S. J. Maybank, and X. Wu, "Human carrying status in visual surveillance," in *Proc. IEEE Comput. Soc. Conf. CVPR*, 2006, vol. 2, pp. 1670–1677.

[11] R. Mehrotra, K. R. Namuduri, and N. Ranganthan, "Gabor filter-based edge detection," *Pattern Recognit.*, vol. 25, no. 12, pp. 1479–1494, Dec. 1992.

[12] F. Pellegrino, W. Vanzella, and V. Torre, "Edge detection revisited," *IEEE Trans. Syst., Man, Cybern.*, vol. 34, no. 3, pp. 1500–1518, Jun. 2004.

[13] W. P. Choi, S. H. Tse, K. W. Wong, and K. M. Lam, "Simplified Gabor wavelets for human face recognition," *Pattern Recognit.*, vol. 41, no. 3, pp. 1186–1199, Mar. 2008.

[14] S. Sardy, "Minimax threshold for denoising complex signals with waveshrink," *IEEE Trans. Signal Process.*, vol. 48, no. 4, pp. 1023–1028, Apr. 2000.

[15] D. Donoho and I. M. Johnstone, *Ideal Spatial Adaptation by Wavelet Shrinkage*. Stanford, CA: Dept. Statist., Stanford Univ., Apr. 1993.

[16] J. Canny, "A computational approach to edge detection," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 1–8, no. 6, pp. 679–698, Nov. 1986.

[17] D. Wang, V. Haese-Coat, A. Bruno, and J. Ronsin, "Some statistical properties of mathematical morphology," *IEEE Trans. Signal Process.*, vol. 43, no. 8, pp. 1955–1965, Aug. 1995.

[18] K. Bowyer, C. Kranenburg, and S. Dougherty, "Edge detector evaluation using empirical ROC curves," *Comput. Vis. Image Underst.*, vol. 84, no. 1, pp. 77–103, Oct. 2001.

[19] K. Woods and K. W. Bowyer, "Generating ROC curves for artificial neural networks," *IEEE Trans. Med. Imag.*, vol. 16, no. 3, pp. 329–337, Jun. 1997.

Wei Jiang received the B.Eng. degree in electronic engineering from Beijing Institute of Technology, Beijing, China, in 2005, where she is currently working toward the Ph.D. degree in electronic engineering in the Department of Electronic Engineering, School of Information Science and Technology.

She is also a Research Assistant with the Centre for Signal Processing, Department of Electronic and Information Engineering, The Hong Kong Polytechnic University, Kowloon, Hong Kong. Her main research interests include image processing, human-face detection, and video tracking.

Kin-Man Lam received the M.Sc. degree in communication engineering from the Department of Electrical Engineering, Imperial College of Science, Technology and Medicine, London, U.K., in 1987 and the Ph.D. degree from the Department of Electrical Engineering, University of Sydney, Sydney, Australia, in 1993.

He received the Associateship in electronic engineering with distinction from The Hong Kong Polytechnic University (formerly called Hong Kong Polytechnic), Kowloon, Hong Kong, in 1986. From 1990 to 1993, he was a Lecturer with the Department of Electronic Engineering, The Hong Kong Polytechnic University, where in October 1996, he was an Assistant Professor with the Centre for Signal Processing, Department of Electronic and Information Engineering, and since February 1999, has been an Associate Professor. His current research interests include human-face recognition, image and video processing, and computer vision.

Dr. Lam was Guest Editor for the Special Issue on Biometric Signal Processing, EURASIP Journal on Applied Signal Processing. He is the Chairman of the IEEE Hong Kong Chapter of Signal Processing and an Associate Editor of EURASIP Journal of Image and Video Processing. He was also a member of the organizing committee and program committee of many international conferences. In particular, he was the Secretary of the 2003 IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP'03), the Technical Chair of the 2004 International Symposium on Intelligent Multimedia, Video and Speech Processing (ISIMP 2004), and a Technical Cochair of the 2005 International Symposium on Intelligent Signal Processing and Communication Systems (ISPACS 2005).

Ting-Zhi Shen received the B.S. degree in electronic engineering and wireless technology from Beijing Institute of Technology, Beijing, China, in 1967.

From 1968 to 1973, she was a Technician with Jinhua Wireless Factory of Instrument Institute, Tianjin, China. Since 1973, she has been with Beijing Institute of Technology, where she is currently a Professor with the Department of Electronic Engineering, School of Information Science and Technology. Her current research interests include human-face recognition, image and video processing, and computer vision.