General Logarithmic Image Processing Convolution

Jose M. Palomares, *Member, IEEE*, Jesús González, Eduardo Ros, and Alberto Prieto, *Senior Member, IEEE*

***Abstract*—The *logarithmic image processing* model (LIP) is a robust mathematical framework, which, among other benefits, behaves invariantly to illumination changes. This paper presents, for the first time, two general formulations of the 2-D convolution of separable kernels under the LIP paradigm. Although both formulations are mathematically equivalent, one of them has been designed avoiding the operations which are computationally expensive in current computers. Therefore, this fast LIP convolution method allows to obtain significant speedups and is more adequate for real-time processing. In order to support these statements, some experimental results are shown in Section V.**

***Index Terms*—Convolution, logarithmic image processing (LIP) average, LIP Gaussian Blur, LIP Sobel.**

I. Introduction

Within the real world, the addition of two images produces a new image which is also visible to the human eye; however, in the digital world the addition of two images may produce "out-of-range" problems, because a value above the saturation threshold (for example, for 8 bits, the maximum value would be 255) is likely to be obtained when two images are digitally added. This value has no meaning for the human visual system, that is, we could not have a value brighter than the absolute brightest or darker than the absolute darkest. In the context of the human brightness perception, there exists a minimum light intensity level, which corresponds to the complete darkness in the human visual system. Another light intensity level, called "upper threshold" or "glare limit," is known [1] to represent the maximum value to which the human visual system is not able to recognize any further increase in the incident light intensity. Saturation of these values to the maximum is often accomplished in digital image processing, but those extreme values are actually never reached in natural images, since our retina, which acts as a natural sensor, works in a logarithmic mode following the *Fechner's Law* [2].

The *logarithmic image processing* (LIP) is a technique initially stated by Pinoli and Jourlin [3], [4] and further developed by Deng *et al.* [5]. All these authors provided a robust mathematical framework with highly desirable properties, such as, the result of the operations is bounded to a specific range. Currently, this paradigm is being expanded to deal with color and not only gray-level images [6], also to use wavelet transforms [7], among other tasks. This framework follows many laws of the human vision system, both physical and psychophysical (among others, *Fechner's law* [8], *Weber's law* [1], [9], *psychophysical contrast notion* [10], *brightness scale inversion* [11], etc.). It is also consistent with the multiplicative transmitted/reflected image formation model [5], sharing the same mathematical basis as the *multiplicative homomorphic image processing* [12] in which the *homomorphic filtering* [13] is based; however, LIP provides operators and a complete framework, at the cost of reformulating the usual image processing techniques. According to [2], LIP has been demonstrated to be a very appropriated model for many different tasks, because it is at the same time mathematically well-justified, physically consistent, psychophysically coherent with higher primates visual system, and computationally affordable. Nowadays, LIP is a mature paradigm used as a tool in applied tasks, such as, face features segmentation [14], among others. Besides, it has become a referenced method to compare with [15], [16].

The convolution is a mathematical tool, which makes use of the neighborhood of each pixel, and, therefore, new images are obtained taking into account context (causal) information. This operator is highly time consuming and data dependant. Furthermore, most of the time, its use becomes a bottleneck in the systems, degrading their data throughput. Many techniques are based on the convolution operator: specially those involving low-pass, band-pass, or high-pass (oriented or not oriented) filtering and their applications such as contrast enhancement [17], [18], both motion and spatial blurring [19]–[21], contrast stretching [22], unsharp masking [22]–[24], smoothing filtering [22], noise elimination [25], [26], *scale-space* filtering [27], [28], etc. Deng and Pinoli [29] showed the effectiveness and applicability of LIP to the edge detection field. They used LIP to state an operator called *LIP Sobel*. Although they did not state explicitly the procedure to create any generic filter but just that specific one, it is easy to extend this methodology to the creation of other filters using the LIP paradigm. This is the main contribution of this work. Our aim is to provide a general formulation of a convolution under the LIP model for image processing, which has not been stated so far. We will focus specifically on the 2-D convolutions which make use of separable kernels because these are the most widely used operators in the image processing field.

This paper is structured as follows. First, in Section II, we will state the definitions used in the article. In Section III, the fundamentals of the LIP paradigm are shown. The formulation of a 2-D convolution with separable kernels under LIP paradigm is stated in Section IV. The LIP convolution is applied to three different applications in Section V. Some illustrative experiments are also described, emphasizing the results in terms of time consumption. Some future work and further applied systems are proposed in Section VI, and, finally, Section VII summarizes the conclusions.

II. Definitions and Ranges

First, we will specify the meaning and the range values of the variables and functions used along this work. M stands for the (unreachable) maximum value allowed depending on the bit depth of the palette used, while 0 is the (unreachable) minimum value allowed for the images. In real-world images, the absolute darkness and the absolute brightness do never occur. If any pixels are found with these extreme values, they are likely to have been obtained by digital operations and/or quantization, and, thus, a shift to the nearest valid value would be affordable without a lack of generality. Gray-level images are notated as capitalized letters such as I, J, etc. The gray-level images used within this article are ranged $I \in (0, M) \subseteq \mathbb{N}$. General gray-level functions are notated as lower case letters, such as f and g, which are two gray-level functions valued $f, g \in (0, M) \subseteq \mathbb{R}$. In general, anything stated for gray-level functions holds true for gray-level images, so we use them indifferently. The *gray tone* of f is notated as the respective letters with a hat, such as $\widehat{f} = M - f$, $\widehat{f} \in (0, M) \subseteq \mathbb{R}$. However, in order to be able to work in an algebraic vector space, the range has been extended to $\widehat{f} \in (-\infty, M) \subseteq \mathbb{R}$, although just the positive *gray tones* physically correspond to bounded

Manuscript received May 3, 2005; revised April 6, 2006. This work was supported in part by the Spanish Ministry of Science and Technology under project DEPROVI (DPI2004-07032). The associate editor coordinating the review of this manuscript and approving it for publication was Dr. Mario A. T. (G. E.) Figueiredo.

J. M. Palomares is with the Department Computer Architecture and Electronics, Escuela Politécnica Superior, University of Cordoba, E.14071, Córdoba, Spain (e-mail: jmpalomares@uco.es).

J. Gonzalez, E. Ros, and A. Prieto are with the Department of Computer Architecture and Computer Technology, E.T.S. Ingeniero Informática y de Telecomunicaciones, University of Granada, E.18071, Granada, Spain (e-mail: jesus@atc.ugr.es; eduardo@atc.ugr.es; aprieto@ugr.es).

Digital Object Identifier 10.1109/TIP.2006.881967

1057-7149/$20.00 © 2006 IEEE

intensity images, while $(-\infty, 0)$ has only a mathematical meaning. For further details, see [29]. The LIP transformed *gray-tone functions* are notated with the respective letters with a tilde, i.e., $\tilde{f} \in \mathbb{R}$. Any scalar is notated as lower case Greek letters, such as, $\alpha \in \mathbb{R}$. Vectors of real scalars are notated in lower case bold face letters, such as $\boldsymbol{a}, \boldsymbol{b}$.

III. Brief Outline of LIP

Although there are several logarithmic models for image processing [2], [12], [30], we focus our work on LIP, which was developed to be able to answer properly to the following question: "*How is the addition of two transmitted images done by the human vision system avoiding the out-of-range problem?*" The authors obtained an appropriate solution following many physical, psychophysical and human perception system laws. There are two strategies to apply LIP philosophy to any image processing technique:

- using the "original" images with some special operators;
- using "transformed" images with the usual operators.

In the following, we include a brief introduction to each of these two methods.

A. First Method: "Original" Images and "Special" Operators

The first option is produced by means of an algebraic vectorial space defined by the following.

- A set of image values (usually named, *gray-tone functions* or simply *gray tone*) which are the "usual" images with an inversion of the scale ($\widehat{I} = M - I$).
- A special sum operator, $\mathbin{\triangle}$ defined as

$$\widehat{f} \mathbin{\triangle} \widehat{g} = \widehat{f} + \widehat{g} - \frac{\widehat{f} \cdot \widehat{g}}{M}. \tag{1}$$

- A special scalar multiplication operator $\mathbin{\blacktriangle}$ computed as

$$\alpha \mathbin{\blacktriangle} \widehat{f} = M - M \cdot \left(1 - \frac{\widehat{f}}{M}\right)^{\alpha}. \tag{2}$$

- Finally, in order to extend the algebraic structure provided by the set of the *gray-tone functions* with the $\mathbin{\triangle}$ and $\mathbin{\blacktriangle}$ operators to a vector space, the negative of each *gray tone* is stated. The subtraction could be defined as in (3), although this operator would produce a *gray-tone function* with no meaning in the real world[1]

$$\widehat{f} \mathbin{\blacktriangledown} \widehat{g} = M \cdot \frac{\widehat{f} - \widehat{g}}{M - \widehat{g}}. \tag{3}$$

Further operators have been proposed, for example, a *LIP Summation* has been defined as

$$\mathop{\triangle}_{i=1}^{n} \widehat{f}_i = \widehat{f}_1 \mathbin{\triangle} \widehat{f}_2 \mathbin{\triangle} \ldots \mathbin{\triangle} \widehat{f_n}. \tag{4}$$

B. Second Method: "Transformed" Images and "Usual" Operators

The second option is to transform the image, working with the "usual" operators and finally, restoring the resulting image to the original space by the inverse of the transforming function.

[1]A negative *gray-tone function* would mean a point in the filter which does not reduce the amount of incident light but rather produces a point which is brighter than the brightest value. This operator has been introduced just to obtain the benefits of an algebraic vector space.

The transform is carried out by using a function called *isomorphic transformation* defined by

$$\tilde{f} = \varphi(\widehat{f}) = -M \cdot \ln\left(1 - \frac{\widehat{f}}{M}\right). \tag{5}$$

The inverse of the function above is called *inverse isomorphic transformation*

$$\widehat{f} = \varphi^{-1}(\tilde{f}) = M \cdot \left(1 - e^{-\frac{\tilde{f}}{M}}\right). \tag{6}$$

IV. LIP 2-D Convolution With Separable Kernels

The convolution operator involves two operands: the signal to be convolved (in our case, the image) and a mask or filter which is the convolving element. If the filter can be expressed by a multiplication of two vectors, it is called *separable*. The use of separable filters in the convolution operator allows to apply two consecutive 1-D convolutions instead of one 2-D convolution [31]. We will focus on a convolution with a separable kernel; that is, given a 2-D filter, F, which is separable $F = \boldsymbol{a}^T \times \boldsymbol{b}$. Let $\boldsymbol{a}$ and $\boldsymbol{b}$ be two row vectors, then

$$\begin{aligned} &\mathrm{conv2D}(I, F) \\ &\quad = \mathrm{conv1D}\left(\mathrm{conv1D}(I, \boldsymbol{a}^T), \boldsymbol{b}\right) \\ &\quad = \sum_{i=0}^{n-1} \boldsymbol{b}(n-i) \cdot \left(\sum_{j=0}^{m-1} \boldsymbol{a}(m-j) \cdot I(i+1, j+1)\right). \end{aligned} \tag{7}$$

We want to compute the LIP version of the 2-D convolution of (7)

$$\begin{aligned} \mathrm{conv2D}_{\triangle}(\widehat{I}, F) = &\mathop{\triangle}_{i=0}^{n-1} \boldsymbol{b}(n-i) \\ &\mathbin{\blacktriangle}\left(\mathop{\triangle}_{j=0}^{m-1} \boldsymbol{a}(m-j) \mathbin{\blacktriangle} \widehat{I}(i+1, j+1)\right). \end{aligned} \tag{8}$$

Let us define $K = \left(\sum_{j=0}^{m-1} \boldsymbol{a}(m-j) \cdot \sum_{i=0}^{n-1} \boldsymbol{b}(n-i)\right)$. Renaming $(M - \widehat{I}(i+1, j+1))$ as $I(i+1, j+1)$ and using the isomorphic transform shown in (5), we obtain

$$\begin{aligned} \varphi\left(\mathrm{conv2D}_{\triangle}(\widehat{I}, F)\right) = -M \cdot \Bigg[&\sum_{i=0}^{n-1} \boldsymbol{b}(n-i) \\ &\cdot \left(\sum_{j=0}^{m-1} \boldsymbol{a}(m-j) \cdot \ln\left(I(i+1, j+1)\right)\right) - K \cdot \ln M\Bigg]. \end{aligned} \tag{9}$$

A. Direct General LIP Convolution: DGLIP-Conv

In the following, a new operator called *DGLIP-Conv* is presented. It follows the first approach to LIP (which we have called the *Direct* method): it works with the "original" images with special operators. It can be easily followed from (9), which may be reformulated as

$$\begin{aligned} \varphi\left(\mathrm{conv2D}_{\triangle}(\widehat{I}, F)\right) = M \cdot \Bigg[&K \cdot \ln M - \\ &\ln\left(\prod_{i=0}^{n-1}\left(\prod_{j=0}^{m-1} I(i+1, j+1)^{\boldsymbol{a}(m-j)}\right)^{\boldsymbol{b}(n-i)}\right)\Bigg]. \end{aligned} \tag{10}$$

Finally, by applying the inverse isomorphic transformation to (10), we obtain

$$\begin{aligned} \mathrm{conv2D}_{\triangle}(\widehat{I}, F) &= \varphi^{-1}\left(\varphi\left(\mathrm{conv2D}_{\triangle}(\widehat{I}, F)\right)\right) \\ &= M \cdot \left(1 - \frac{\prod_{i=0}^{n-1}\prod_{j=0}^{m-1}\left(I(i+1, j+1)^{\boldsymbol{a}(m-j)}\right)^{\boldsymbol{b}(n-i)}}{M^K}\right). \end{aligned} \tag{11}$$

B. Fast General LIP Convolution: FGLIP-Conv

The computation of the *LIP Convolution* with real time constraints is challenging and can highly benefit from this alternative strategy we present in this section. The previously stated *DGLIP-Conv* makes use of divisions and exponentiations, which are computationally expensive operations. Therefore, a new formulation of the operator following the other LIP approach (working with "transformed" images and usual operators) has been developed. This formulation avoids these time-consuming instructions, while obtaining the same results than *DGLIP-Conv*, and requiring much less computing resources. Using previously stated K, (9) can be rewritten as

$$\varphi\left(\mathrm{conv2D}_{\Delta}(\widehat{I},F)\right) = M\cdot\left(K\cdot\ln M-\mathrm{conv1D}\left(\mathrm{conv1D}(\ln I,\boldsymbol{a}^{T}),\boldsymbol{b}\right)\right). \quad (12)$$

Finally, applying the inverse isomorphic transformation (6)–(12)

$$\mathrm{conv2D}_{\Delta}(\widehat{I},F) = \varphi^{-1}\left(\varphi\left(\mathrm{conv2D}_{\Delta}(\widehat{I},F)\right)\right) = M\cdot\left(1-e^{\mathrm{conv\,1D}\left(\mathrm{conv\,1D}(\ln I,\boldsymbol{a}^{T}),\boldsymbol{b}\right)-K\cdot\ln M}\right). \quad (13)$$

It is worth remarking that in both (11) and (13), the original image, and not the *gray-tone function* image, is processed. Therefore, we avoid one preprocessing step (the subtraction to obtain the *gray-tone function* is, hence, useless).

V. Application of the LIP Convolution

After having described two formulations of the convolution under LIP paradigm, we evaluate their effectiveness in three different applications within the image processing field. This illustrates how easy the customization of the 2-D LIP paradigm to concrete filters becomes. In (13), the $\ln I$ can be computed with a lookup table (LUT), because only $M-2$ integer values, ranging from 1 to $M-1$, are involved. This has not been applied in the following experiments, and, thus, the natural logarithm has been calculated for each pixel in the image. In this way, we show that the *FGLIP-Conv* is faster, regardless of the implementation of the logarithmic operation. First, the *Sobel* edge detector is redefined under the LIP paradigm. After that, two low-pass filtering experiments, by means of the averaging within a neighborhood for two different sizes of the neighborhood, are stated. And we finally include experiments with Gaussian blurring. All the experiments are applied to two sizes of images: 512 × 512 and 320 × 240.

A. LIP Sobel

Deng and Pinoli proposed a reformulation of the well-known *Sobel* method using the LIP paradigm [29]. In that case, the contents of filters to be applied were known in advance, and, thus, authors could design an adapted formula, which, however, was not general, but specific to that concrete task. This new method detected edges either in well or poorly lit areas of intensity images, allowing the detectors to be robust and almost illumination invariant. A brief description of the method is included here. Any given 3 × 3 area of a discrete *gray-tone function* is notated as $\widehat{f}_1,\widehat{f}_2\ldots\widehat{f}_9$, for each *gray-tone function* pixel from left to right and from top to bottom. Using that neighborhood, authors stated the *LIP Sobel gray-tone* vector, $\widehat{\boldsymbol{g}}=(\widehat{g_x},\widehat{g_y})$, given by

$$\widehat{g_x} = \left(\widehat{f}_1 \mathbin{\mathbb{\Delta}} (2\mathbin{\mathbb{\Delta}}\widehat{f}_4)\mathbin{\mathbb{\Delta}}\widehat{f}_7\right)\mathbin{\mathbb{\Delta}}\left(\widehat{f}_3\mathbin{\mathbb{\Delta}}(2\mathbin{\mathbb{\Delta}}\widehat{f}_6)\mathbin{\mathbb{\Delta}}\widehat{f}_9\right) \quad (14)$$

$$\widehat{g_y} = \left(\widehat{f}_1 \mathbin{\mathbb{\Delta}} (2\mathbin{\mathbb{\Delta}}\widehat{f}_2)\mathbin{\mathbb{\Delta}}\widehat{f}_3\right)\mathbin{\mathbb{\Delta}}\left(\widehat{f}_7\mathbin{\mathbb{\Delta}}(2\mathbin{\mathbb{\Delta}}\widehat{f}_8)\mathbin{\mathbb{\Delta}}\widehat{f}_9\right). \quad (15)$$

These authors were able to define the *LIP Sobel* vector because they knew, in advance, the values of the filter (a negative or positive value, which was translated into the usage of $\mathbb{\Delta}$ or $\mathbb{\Delta}$, respectively). Using the LIP isomorphism $\varphi(\cdot)$ in (5)

$$\widehat{g_x} = M - M\left(\frac{f_3 f_6^2 f_9}{f_1 f_4^2 f_7}\right). \quad \widehat{g_y} = M - M\left(\frac{f_1 f_2^2 f_3}{f_7 f_8^2 f_9}\right). \quad (16)$$

After that, a gradient map image can be calculated using

$$\mathrm{grad}_{\Delta}(\widehat{\boldsymbol{g}}) = M\cdot\left(1-\exp\left(-\frac{\sqrt{\varphi(\widehat{g_x})^2+\varphi(\widehat{g_y})^2}}{M}\right)\right) \quad (17)$$

as it was done in [29]. As it can be easily deduced, (16) can be directly obtained from (11), and, thus, *DGLIP-Conv* is a more general formulation. It is obvious that, *FGLIP-Conv*, described in (13), shows a general formulation as well. Both of them include Deng and Pinoli contribution [29] as a particular case.

Fig. 1. Original "peppers" image.

Fig. 2. Darkened "peppers" image.

To test the *LIP Sobel* method, the well-known "peppers" image (shown in Fig. 1) has been used. Fig. 2 shows the result of darkening the original image (Fig. 1) with the darkening formula

$$D(x,y) = \left\lceil I(x,y)\cdot\left(0.1+\frac{5\cdot\sin\left(\frac{\pi}{2}\cdot\frac{x}{\mathrm{width}}\right)}{6}\right)\right\rceil \quad (18)$$

where "width" is the width of the image to be darkened and $I(x,y)$ is the value of the (x,y) pixel of the original image. In the following, we compare the results of the *LIP Sobel* by the three different methods (Deng and Pinoli, *DGLIP-Conv*, and *FGLIP-Conv*). The results obtained by the three methods are exactly the same.

1) LIP Sobel by Deng and Pinoli's Method: The standard Sobel applied on the darkened image is shown in Fig. 3. The *LIP Sobel* method

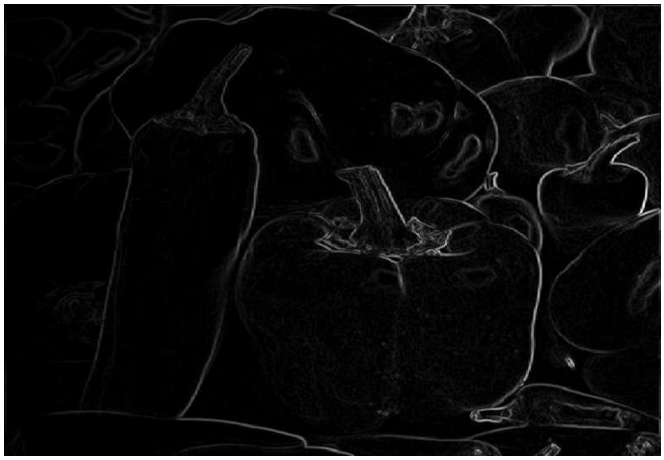

Fig. 3. Standard Sobel applied on darkened "peppers" image.

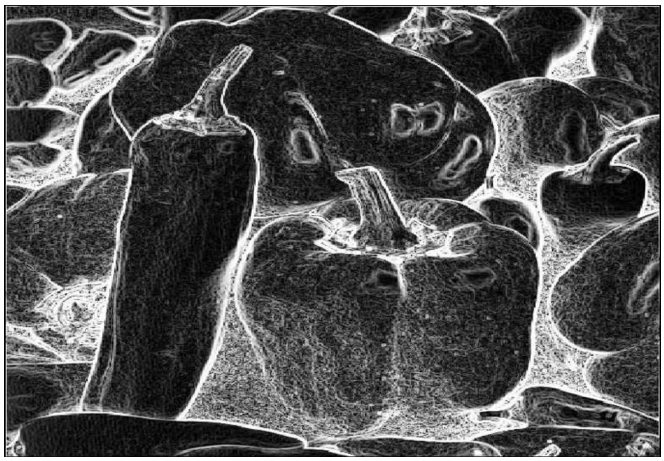

Fig. 4. *LIP Sobel* applied on darkened "peppers" image.

as proposed by Deng and Pinoli, stated in (16), is applied on Fig. 2 and the resulting image is shown in Fig. 4. It is worth remarking that the edges in the dark regions are not clearly detected in Fig. 3; however, in Fig. 4, the edges in the dark regions are more homogenously detected. This is a beneficial collateral feature which is a characteristic of the LIP approach.

2) LIP Sobel by the LIP Convolution Methods: The standard Sobel filter is separable. It can be obtained with two vectors, $\boldsymbol{a} = [-1, 0, 1]$ and $\boldsymbol{b} = [1, 2, 1]$. In order to obtain the *FGLIP-Conv*, we applied (13), using $\boldsymbol{a}$ and $\boldsymbol{b}$ as stated previously, to the darkened image (Fig. 2). In a similar way, the same result is achieved using the *DGLIP-Conv* applying (11) with $\boldsymbol{a}$ and $\boldsymbol{b}$. The edges detected by both these methods are the same as those computed by the method proposed by Deng and Pinoli (see Fig. 4). We conclude that all three methods have obtained the same results, since the mean squared error is negligible (less than 10^{-12}).

B. Parameterized-Size LIP Filtering With a Constant Filter: LIP Average

In the previous application, we have shown that our methods obtain the same results as a specific algorithm. However, the presented general methods provide a versatile and general framework. Following the same procedure as in Deng and Pinoli's *LIP Sobel* to obtain the formulae, we can design (by means of Deng and Pinoli's method) any other filter. However, we should design specific formulae for every different size of the neighborhood. On the other hand, with our proposed general convolution LIP procedure, just by changing the input parameters (1-D filters), we are able to use the same code, resulting in a gain of flexibility and also of robustness. This is one of the main advantages of our general methods, but also their significant speedup, as it will be illustrated in Section V-D.

We now issue the "best" case for the Deng and Pinoli's method in terms of speed, considering a filter in which all elements are constants, for example, an average neighborhood filter (a low-pass filter). This is the "best" case for the Deng and Pinoli's method, because exponentiation, which is a heavy computational operation, is done just once per filter, as we will show in the following paragraph. Besides, we compare that result with those obtained using the *LIP Convolution* methods (both the *DGLIP-Conv* and the *FGLIP-Conv* ones). Again, the resulting images are the same using the three methods.

1) $n \times n$-Neighborhood LIP-Average Filtering by Deng and Pinoli's Method: Let us define an average filtering using LIP model on a $n \times n$-neighborhood. First of all, we should state the value of n in order to be able to define a specific formula. With $n = 3$, and using the same definition of a neighborhood as in the *LIP Sobel*, we obtain

$$\begin{aligned}\operatorname{avg}_{\mathrm{LIP}}^{3\times 3}(f) = \Bigg(\Bigg(\Bigg(\frac{1}{9} \mathbin{\text{⩓}} \widehat{f}_1\Bigg) \mathbin{\text{⩓}} \Bigg(\frac{1}{9} \mathbin{\text{⩓}} \widehat{f}_2\Bigg) \mathbin{\text{⩓}} \Bigg(\frac{1}{9} \mathbin{\text{⩓}} \widehat{f}_3\Bigg) \\ \mathbin{\text{⩓}} \Bigg(\frac{1}{9} \mathbin{\text{⩓}} \widehat{f}_4\Bigg) \mathbin{\text{⩓}} \Bigg(\frac{1}{9} \mathbin{\text{⩓}} \widehat{f}_5\Bigg) \mathbin{\text{⩓}} \Bigg(\frac{1}{9} \mathbin{\text{⩓}} \widehat{f}_6\Bigg) \\ \mathbin{\text{⩓}} \Bigg(\frac{1}{9} \mathbin{\text{⩓}} \widehat{f}_7\Bigg) \mathbin{\text{⩓}} \Bigg(\frac{1}{9} \mathbin{\text{⩓}} \widehat{f}_8\Bigg) \mathbin{\text{⩓}} \Bigg(\frac{1}{9} \mathbin{\text{⩓}} \widehat{f}_9\Bigg)\Bigg). \quad (19)\end{aligned}$$

This leads us to the following formula:

$$\operatorname{avg}_{\mathrm{LIP}}^{3\times 3}(f) = M - (f_1 \cdot f_2 \cdot f_3 \cdot f_4 \cdot f_5 \cdot f_6 \cdot f_7 \cdot f_8 \cdot f_9)^{\frac{1}{9}}. \quad (20)$$

This can be written shortly as

$$\operatorname{avg}_{\mathrm{LIP}}^{3\times 3}(f) = M - \left(\prod_{i=1}^{9} f_i\right)^{\frac{1}{9}}. \quad (21)$$

We are able to take 1/9 as a common exponent, which reduces the complexity of the formula to a great extent.

It has been tested also this experiment for a larger neighborhood: 5× 5. In this case, the formula applied is

$$\operatorname{avg}_{\mathrm{LIP}}^{5\times 5}(f) = M - \left(\prod_{i=1}^{25} f_i\right)^{\frac{1}{25}}. \quad (22)$$

It is clear that for each value of n, that is, for each different size of the neighborhood involved, a new formula is needed, and, thus, it would be necessary to write a new code each time.

2) $n \times n$-Neighborhood LIP Average Filtering by the LIP Convolution Methods: On the other hand, another formulation of the previously stated 3 × 3-neighborhood, a *LIP Average* filtering can be obtained by applying (13), or (11), with vectors $\boldsymbol{a} = \boldsymbol{b} = [1/3, 1/3, 1/3]$.

In order to compute the average in a 5 × 5-neighborhood, we can apply (13), or (11), with vectors $\boldsymbol{a} = \boldsymbol{b} = [1/5, 1/5, 1/5, 1/5, 1/5]$. It can be easily deduced that just by changing the filters that are used as input parameters, we are able to obtain the computation of the average filters with different sizes of the neighborhood, without rewriting any line of code.

C. Parameterized-Size LIP Filtering: LIP Gaussian Blur

In the previous section, we showed a pair of examples of the "best" case, in terms of computational load, for the Deng and Pinoli's method. We affirm that it is "best" case because a reduction of the exponentiations involved could be achieved; as every pixel was raised to the same value, a common exponent could be taken. Therefore, the "worst" case is given by a filter in which we cannot optimize the Deng and

Pinoli's method and a common exponent cannot be taken. This is the case of a filter in which every value is different, for example, a Gaussian filter used in a blurring. We have also chosen this example because the 2-D Gaussian filter is separable. Thus, a usual way of obtaining a 2-D Gaussian filter is by computing the product of a 1-D Gaussian with itself. As a specific example, we have developed a 7×7 2-D Gaussian filter by approximation of the Gaussian (with $\sigma = 1.0$) in a 7×7 mesh.

We include in the next paragraphs the results obtained by the *LIP Gaussian Blur* programmed using the three different methods under consideration. Obviously, as expected, there are no differences between the results obtained by any of the three methods.

1) LIP Gaussian Blur by Deng and Pinoli's Method: We have extended the definition of the neighborhood commented in the *LIP Sobel* section to a 7×7 neighborhood. In the following, we show the formula used to apply a Gaussian blur under LIP paradigm (for the sake of simplicity not all values are presented)

$$\begin{aligned}&\mathrm{GBlur}_{\mathrm{LIP}}(f) = \\ &\Big((0.0001 \mathbin{\text{▲}} \widehat{f_1}) \mathbin{\text{▲}} (0.0015 \mathbin{\text{▲}} \widehat{f_2}) \mathbin{\text{▲}} (0.0067 \mathbin{\text{▲}} \widehat{f_3}) \mathbin{\text{▲}} \ldots \\ &\ldots (0.1353 \mathbin{\text{▲}} \widehat{f_{23}}) \mathbin{\text{▲}} (0.6065 \mathbin{\text{▲}} \widehat{f_{24}}) \mathbin{\text{▲}} (1.0000 \mathbin{\text{▲}} \widehat{f_{25}}) \mathbin{\text{▲}} \ldots \\ &\ldots (0.0067 \mathbin{\text{▲}} \widehat{f_{47}}) \mathbin{\text{▲}} (0.0015 \mathbin{\text{▲}} \widehat{f_{48}}) \mathbin{\text{▲}} (0.0001 \mathbin{\text{▲}} \widehat{f_{49}})\Big). \quad (23)\end{aligned}$$

Analogously to what is done to obtain (16), the same strategy can lead to the following formula:

$$\begin{aligned}&\mathrm{GBlur}_{\mathrm{LIP}}(f) \\ &\quad = M - \left(\frac{f_1^{0.0001} \cdot \ldots \cdot f_{24}^{0.6065} \cdot f_{25}^{1.000} \cdot \ldots \cdot f_{49}^{0.0001}}{M^{0.0001 \cdot \ldots \cdot 0.6065 \cdot 1.0000 \cdot \ldots \cdot 0.0001}}\right). \quad (24)\end{aligned}$$

Opposite to (21) and (22), in this formula, we cannot take any common exponent, and, therefore, every pixel of the image is exponentiated to the value in correspondence in the Gaussian filter.

2) LIP Gaussian Blur by the LIP Convolution Methods: As it has been stated above, a 2-D Gaussian matrix is a separable filter, built up with a multiplication of a 1-D Gaussian with itself. The size of the filter involved depends on the value of the σ parameter: the larger the σ is, the larger the size of the filter. As a rule of thumb, the length of the Gaussian filter that shall be taken, centered on a specific position, is $3.5 \times \sigma^2$ values left and right of that position (rounding down to the lower integer value is acceptable). For example, for $\sigma = 1$, a Gaussian filter with length of 7 shall be taken. For that given σ, the following vectors have been used in the experiment $\boldsymbol{a} = \boldsymbol{b} = [0.011, 0.135, 0.606, 1, 0.606, 0.135, 0.011]$. Of course, for a larger value of σ, a larger filter shall be taken. However, there is no need to rewrite any line of code, just by changing the parameters, the desired output will be produced.

D. Speedup Analysis

In this section, we focus on the computation time spent by each method on each experiment. For this task, we have implemented all the algorithms in MATLAB©7 (R14) using the Image Processing Toolbox. We tried to be as fair as possible; we used the functions recommended by Matlab developers with the data types for which optimizations were enabled—no further direct optimizations were introduced. Using a Pentium Centrino M725 at 1.60 GHz (2-MB L2-Cache) with 512-MB RAM, we produced a set of experiments with two different sizes for the images (one of 512×512 and the other using the standard CIF size of 320×240), for which several executions were tried, taking the average of the time obtained for each individual execution (see Table I).

The execution times cannot be compared for each experiment and method, because the figures are considerably different. However, if we consider the results of the experiments using the Deng and Pinoli's method as the base reference time, we can compute the gain of speed of each method for every experiment compared to that base reference time. This gain of speed, or "speedup," describes the relative speed of a new system in relation to another system, which is taken as the reference one. If the speedup value is above one, it means that the new system is faster than the reference one; while, if the speedup is lower than one, it means that the new system is slower than the reference one.

TABLE I
COMPUTATION TIME AND SPEEDUP COMPARISON (TIME IN SECONDS)

Application	Size	Method	Time	Speedup
LIP-Sobel	512×512	Deng & Pinoli	1.008	1.00
		DGLIP–Conv	5.847	0.17
		FGLIP–Conv	0.450	2.24
	320×240	Deng & Pinoli	0.386	1.00
		DGLIP–Conv	1.682	0.23
		FGLIP–Conv	0.155	2.49
LIP–Average (3×3)	512×512	Deng & Pinoli	0.456	1.00
		DGLIP–Conv	3.448	0.13
		FGLIP–Conv	0.192	2.38
	320×240	Deng & Pinoli	0.227	1.00
		DGLIP–Conv	1.017	0.22
		FGLIP–Conv	0.076	2.99
LIP–Average (5×5)	512×512	Deng & Pinoli	0.482	1.00
		DGLIP–Conv	4.056	0.12
		FGLIP–Conv	0.198	2.43
	320×240	Deng & Pinoli	0.246	1.00
		DGLIP–Conv	1.189	0.21
		FGLIP–Conv	0.077	3.20
LIP–Gaussian Blur	512×512	Deng & Pinoli	9.629	1.00
		DGLIP–Conv	4.560	2.11
		FGLIP–Conv	0.333	28.92
	320×240	Deng & Pinoli	2.855	1.00
		DGLIP–Conv	1.421	2.01
		FGLIP–Conv	0.175	16.31

The speedup of every method and application is shown in Table I. After computing the speedup, we can infer that the *FGLIP-Conv* is much faster than the Deng and Pinoli's method in every application and size of the image. The mean speedup of the *FGLIP-Conv* is approximately 2.5 faster[2] than the Deng and Pinoli's method. One of the reasons for this speedup is due to the use of separable kernels that this new formulation allows, in opposition to the Deng and Pinoli's method, in which there does not exist a technique to make use of the separability of the filters. Another reason for the speedup is that the *FGLIP-Conv* makes use of additions and multiplications, opposed to the Deng and Pinoli's method, which is based on multiplications and exponentiations, much slower and computationally more complex than the first ones. On the other hand, the *DGLIP-Conv* is slower than the Deng and Pinoli's method almost in every experiment. This is because although both the *DGLIP-Conv* and the Deng and Pinoli's methods use multiplications and exponentiations, the Deng and Pinoli's method is optimized, reducing the amount of exponentiations involved, while *DGLIP-Conv* is general. However, there exists a set of experiments, the *LIP Gaussian Blur*, for which even the *DGLIP-Conv* is twice faster. In that case, the Deng and Pinoli's method cannot make a reduction of the exponentiations, one for each element in the filter (for that method, it is a 2-D filter) in opposition to the *DGLIP-Conv* in which there is also one exponentiation per element in the filter, but with two separable 1-D filters. We can infer that the usage of separable kernels rises the speed to twice faster; thus, we are able to affirm that any amount above that number is due only to the lower complexity of the operations performed. Besides, for that set of experiments, the speedup of the

[2]In fact, the average speedup of the *FGLIP-Conv* is 7.62. However, if we do not take into account a pair of extreme values (those obtained in the *LIP Gaussian Blur* experiments), a speedup of 2.62 is computed.

FGLIP-Conv is almost thirty times faster. This very significant speedup is due to the use by the *FGLIP-Conv* of additions and multiplications and a 1-D filter, against the use of multiplications and exponentiations and a 2-D filter.

VI. Future Work

These two methods, particularly the *FGLIP-Conv*, open up a great range of possibilities in the fields of image and video processing. Currently, we are considering to design this operator making use of the SIMD enhancement included in the Intel processors (that is, MMX/SSE$*$) jointly with C/C++ coding. This is expected to make this operator even faster on this family of processors. It will also allow us to carry out a generic study on how to parallelize it, which will allow us to port it easily to other SIMD architectures.

Using this general operator, other more elaborated techniques will be easier to obtain. For example, the building up of a *LIP Canny* edge detector operator would be very interesting, trying to make it almost invariant to slow changes of illuminance, as the *LIP Sobel* does. This operator represents another step towards the first stage of a complex human-like visual understanding system: a robust and fast technique which has all the LIP advantages and is also illumination invariant. This beneficial property makes useless to perform a *Homomorphic Filtering* on the images to obtain the reflectance images of each one, which will be translated in a lower execution time.

VII. Conclusion

Taking into account that convolution is a very useful and widely used tool in the fields of image and video processing, and that LIP is a very robust mathematical framework with different interesting properties, we have combined in this work these two techniques into a general formulation of the convolution under the LIP paradigm. It is worth remarking that, up to the current article, the convolution operator had not been stated using the LIP model. In the present work, two new formulations of a 2-D convolution under the LIP paradigm, *FGLIP-Conv* and *DGLIP-Conv*, have been described. These two formulations have been designed generically: any 2-D separable kernel of any size can be applied. *FGLIP-Conv* and *DGLIP-Conv* represent a general framework in which new 2-D convolution filters can be easily defined.

Three different applications have been chosen to evaluate the scheme, using different values for the size of the images. It has been shown that the new methods lead to the same results as the method proposed by Deng and Pinoli. The original method (called here Deng and Pinoli's method) has been programmed using a matrix-based processing, instead of an element by element iterative processing, which means a significative speedup. However, the computation times of the experiments show that the *FGLIP-Conv*, proposed in this work, is the best choice for applications which are sensitive to the processing time. The experiment which is considered to be the "best" case for the Deng and Pinoli's method is approximately 60% slower than the *FGLIP-Conv*, while in the experiment which would be among the "worst" cases, the Deng and Pinoli's method is 96.5% slower than the *FGLIP-Conv*. These results are due to the use of multiplications and additions instead of multiplications and exponentiations, which are much slower.

Acknowledgment

The authors would like to thank the reviewers and the Associate Editor for their valuable suggestions. They would also like to thank A. M. Palomares, F. J. Vigier, and M. McManus for their translation of the original manuscript.

References

[1] I. E. Gordon, *Theories of Visual Perception*, 3rd ed. Hove, U.K.: Psychology Press, 2004.
[2] J. C. Pinoli, "A general comparative study of the multiplicative homomorphic, log-ratio and logarithmic image processing approaches," *Signal Process.*, vol. 58, no. 1, pp. 11–45, Apr. 1997.
[3] M. Jourlin and J. C. Pinoli, "Logarithmic image processing," *Acta Stereol.*, vol. 6, pp. 651–656, 1987.
[4] ——, "A model for logarithmic image-processing," *J. Microsc.*, vol. 149, pp. 21–35, Jan. 1988.
[5] G. Deng, L. W. Cahill, and G. R. Tobin, "The study of logarithmic image-processing model and its application to image-enhancement," *IEEE Trans. Image Process.*, vol. 4, no. 4, pp. 506–512, Apr. 1995.
[6] V. Patrascu, "Color image enhancement using the support fuzzification," in *Proc. 10th IFSA*, Istanbul, Turkey, Jul. 2003, pp. 412–419.
[7] G. Courbebaisse, F. Trunde, and M. Jourlin, "Wavelet transform and LIP model," *Image Anal. Stereol.*, vol. 21, no. 2, pp. 121–125, Jun. 2002.
[8] L. W. Hurvich and D. Jameson, *The Perception of Brightness and Darkness*. Boston, MA: Allyn & Bacon, 1966.
[9] V. Bruni and D. Vitulano, "A generalized model for scratch detection," *IEEE Trans. Image Process.*, vol. 13, no. 1, pp. 44–50, Jan. 2004.
[10] R. Watt, *Understanding Vision*. London, U.K.: Academic, 1991.
[11] D. A. Baylor, T. D. Lamb, and K. W. Yau, "Responses of retinal rods to single photons," *J. Physiol.*, vol. 288, no. 1, pp. 613–634, Mar. 1979.
[12] A. V. Oppenheim, R. W. Schafer, and T. G. Stockham, "Nonlinear filtering of multiplied and convolved signals," *Proc. IEEE*, vol. 56, no. 8, pp. 1264–1291, Aug. 1968.
[13] D. Toth, T. Aach, and V. Metzler, "Illumination-invariant change detection," in *Proc. 4th IEEE SSIAI*, Austin, TX, Apr. 2000, pp. 3–7.
[14] M. Lievin and F. Luthon, "Nonlinear color space and spatiotemporal MRF for hierarchical segmentation of face features in video," *IEEE Trans. Image Process,*, vol. 13, no. 1, pp. 63–71, Jan. 2004.
[15] B. Abidi, J. Liang, M. Mitckes, and M. Abidi, "Improving the detection of low-density weapons in x-ray luggage scans using image enhancement and novel scene-decluttering techniques," *J. Electron. Imag.*, vol. 13, no. 3, pp. 523–538, Jul. 2004.
[16] G. Ramponi, "A cubic unsharp masking technique for contrast enhancement," *Signal Process.*, vol. 67, no. 2, pp. 211–222, June 1998.
[17] W. K. Pratt, *Digital Image Processing*, 2nd ed. New York: Wiley, 1991.
[18] R. C. Gonzalez and R. E. Woods, *Digital Image Processing*, 2nd ed. Upper Saddle River, NJ: Prentice-Hall, 2002.
[19] D. Marr and E. Hildreth, "Theory of edge-detection," *Proc. Roy. Soc. Lond. Ser. B*, vol. 207, no. 1167, pp. 187–217, 1980.
[20] M. Potmesil and I. Chakravarty, "Modeling motion blur in computer-generated images," in *Proc. 10th Annu. Conf. Computer Graphics Interac. Tech.*, Detroit, MI, Jul. 1983, pp. 389–399.
[21] G. J. Brostow and I. Essa, "Image-based motion blur for stop motion animation," in *Proc. 28th Annu. Conf. SIGGRAPH*, Los Angeles, CA, Aug. 2001, pp. 561–566.
[22] A. K. Jain, *Fundamentals of Digital Image Processing*. Englewood Cliffs, NJ: Prentice-Hall, 1989.
[23] P. G. Powell and B. E. Bayer, "A method for the digital enhancement of unsharp, grainy photographic images," in *Proc. IEEE Int. Conf. Electronic Image Processing*, Jul. 1982, pp. 179–183.
[24] B. Horn, *Robot Vision*. Cambridge, MA: MIT Press, 1986.
[25] A. Bovik, Ed., *Handbook of Image and Video Processing*. London, U.K.: Academic, 2000.
[26] X. Xu, E. Miller, D. Chen, and M. Sarhadi, "Adaptive two-pass rank order filter to remove impulse noise in highly corrupted images," *IEEE Trans. Image Process.*, vol. 13, no. 2, pp. 238–247, Feb. 2004.
[27] A. Witkin, "Scale-space filtering," in *Proc. IJCAI*, Karlsruhe, Germany, 1983, pp. 1019–1022.
[28] A. Petrovic, O. Divorra Escoda, and P. Vandergheynst, "Multiresolution segmentation of natural images: from linear to nonlinear scale-space representations," *IEEE Trans. Image Process.*, vol. 13, no. 8, pp. 1104–1114, Aug. 2004.

[29] G. Deng and J. C. Pinoli, "Differentiation-based edge detection using the logarithmic image processing model," *J. Math. Imag. Vis.*, vol. 8, no. 2, pp. 161–180, Mar. 1998.
[30] V. Patrascu and V. Buzuloiu, "Image dynamic range enhancement in the context of logarithmic models," in *Proc. 11th Eur. Signal Process.*, Toulouse, France, Sep. 2002, vol. 3, pp. 251–254.
[31] W. Rudin, *Real and Complex Analysis (Higher Mathematics Series)*, 3rd ed. New York: McGraw-Hill, 1986.

Orthogonal Laplacianfaces for Face Recognition

Deng Cai, Xiaofei He, Jiawei Han, *Senior Member, IEEE*, and Hong-Jiang Zhang, *Fellow, IEEE*

***Abstract*—Following the intuition that the naturally occurring face data may be generated by sampling a probability distribution that has support on or near a submanifold of ambient space, we propose an appearance-based face recognition method, called orthogonal Laplacianface. Our algorithm is based on the locality preserving projection (LPP) algorithm, which aims at finding a linear approximation to the eigenfunctions of the Laplace Beltrami operator on the face manifold. However, LPP is nonorthogonal, and this makes it difficult to reconstruct the data. The orthogonal locality preserving projection (OLPP) method produces orthogonal basis functions and can have more locality preserving power than LPP. Since the locality preserving power is potentially related to the discriminating power, the OLPP is expected to have more discriminating power than LPP. Experimental results on three face databases demonstrate the effectiveness of our proposed algorithm.**

***Index Terms*—Appearance-based vision, face recognition, locality preserving projection (LPP), orthogonal locality preserving projection (OLPP).**

I. Introduction

Recently, appearance-based face recognition has received a lot of attention [20], [14]. In general, a face image of size $n_1 \times n_2$ is represented as a vector in the image space $\mathbb{R}^{n_1 \times n_2}$. We denote by face space the set of all the face images. Though the image space is very high dimensional, the face space is usually a submanifold of very low dimensionality which is embedded in the ambient space. A common way to attempt to resolve this problem is to use dimensionality reduction techniques [1], [2], [8], [11], [12], [17]. The most popular methods discovering the face manifold structure include Eigenface [20], Fisherface [2], and Laplacianface [9].

Face representation is fundamentally related to the problem of manifold learning [3], [16], [19] which is an emerging research area. Given a set of high-dimensional data points, manifold learning techniques aim at discovering the geometric properties of the data space, such as its Euclidean embedding, intrinsic dimensionality, connected components, homology, etc. Particularly, learning representation is closely related to the embedding problem, while clustering can be thought of as finding connected components. Finding a Euclidean embedding of the face space for recognition is the primary focus of our work in this paper. Manifold learning techniques can be classified into linear and nonlinear techniques. For face processing, we are especially interested in linear techniques due to the consideration of computational complexity.

The Eigenface and Fisherface methods are two of the most popular linear techniques for face recognition. Eigenface applies Principal Component Analysis (PCA) [6] to project the data points along the directions of maximal variances. The Eigenface method is guaranteed to discover the intrinsic geometry of the face manifold when it is linear. Unlike the Eigenface method which is unsupervised, the Fisherface method is supervised. Fisherface applies Linear Discriminant Analysis (LDA) to project the data points along the directions optimal for discrimination. Both Eigenface and Fisherface see only the global Euclidean structure. The Laplacianface method [9] is recently proposed to model the local manifold structure. The Laplacianfaces are the linear approximations to the eigenfunctions of the Laplace Beltrami operator on the face manifold. However, the basis functions obtained by the Laplacianface method are nonorthogonal. This makes it difficult to reconstruct the data.

In this paper, we propose a new algorithm called **orthogonal Laplacianface**. O-Laplacianface is fundamentally based on the Laplacianface method. It builds an adjacency graph which can best reflect the geometry of the face manifold and the class relationship between the sample points. The projections are then obtained by preserving such a graph structure. It shares the same locality preserving character as Laplacianface, but at the same time it requires the basis functions to be orthogonal. Orthogonal basis functions preserve the metric structure of the face space. In fact, if we use all the dimensions obtained by O-Laplacianface, the projective map is simply a rotation map which does not distort the metric structure. Moreover, our empirical study shows that O-Laplacianface can have more locality preserving power than Laplacianface. Since it has been shown that the locality preserving power is directly related to the discriminating power [9], the O-Laplacianface is expected to have more discriminating power than Laplacianface.

The rest of the paper is organized as follows. In Section II, we give a brief review of the Laplacianface algorithm. Section III introduces our O-Laplacianface algorithm. We provide a theoretical justification of our algorithm in Section IV. Extensive experimental results on face recognition are presented in Section V. Finally, we provide some concluding remarks and suggestions for future work in Section VI.

II. Brief Review of Laplacianface

Laplacianface is a recently proposed linear method for face representation and recognition. It is based on locality preserving projection [10] and explicitly considers the manifold structure of the face space.

Given a set of face images $\{\mathbf{x}_1, \ldots, \mathbf{x}_n\} \subset \mathbb{R}^m$, let $X = [\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_n]$. Let S be a similarity matrix defined on the data points. Laplacianface can be obtained by solving the following minimization problem:

$$\begin{aligned}\mathbf{a}_{opt} &= \arg\min_{\mathbf{a}} \sum_{i=1}^{n}\sum_{j=1}^{n}\left(\mathbf{a}^T\mathbf{x}_i - \mathbf{a}^T\mathbf{x}_j\right)^2 S_{ij} \\ &= \arg\min_{\mathbf{a}} \mathbf{a}^T X L X^T \mathbf{a}\end{aligned}$$

Manuscript received August 30, 2005; revised March 22, 2006. The associate editor coordinating the review of this manuscript and approving it for publication was Dr. Manuel Sameulides.

D. Cai and J. Han are with the Department of Computer Science, University of Illinois at Urbana Champaign, Urbana, IL 61801 USA (e-mail: dengcai2@cs.uiuc.edu; hanj@cs.uiuc.edu).
X. He is with Yahoo Research Labs, Burbank, CA 91504 USA (e-mail: hex@yahoo-inc.com).
H.-J. Zhang is with Microsoft Research Asia, Beijing 100080, China (e-mail: hjzhang@microsoft.com).

Color versions of Figs. 1 and 3–5 are available online at http://ieeexplore.ieee.org.
Digital Object Identifier 10.1109/TIP.2006.881945

1057-7149/$20.00 © 2006 IEEE