二维滑动矩形窗傅氏变换的快速递推算法 [1]

张丽飞 杨鸿波 王东峰 邹谋炎

(中国科学院电子学研究所 北京 100080)

摘 要 该文利用相邻滑动窗数据之间的关系以及傅氏变换的平移性质，提出一种二维滑动矩形窗傅氏变换的快速递推算法．文中分析了该快速递推算法的复杂度和传统直接计算法的复杂度，证明了新的快速递推法可以大大降低计算复杂性，尤其是在图像尺寸和窗口尺寸较大的场合中．该算法可以改善滑窗傅氏变换或 Gabor 变换的计算效率．

关键词 图像分析，纹理分析，快速傅氏变换，Gabor 变换，递推算法

中图号 TN911.72, O174.2

1 引 言

图像分析，特别是图像纹理分析和图像分割，常常需计算图像中以各像素点为中心的一个邻域上纹理的特征量．当各像素点所在邻域属于同一种纹理时，纹理特征量的特性基本一致；否则，特征量的特性差别较大．提取纹理特征的算法一方面必须给出有意义特征量，另一方面算法复杂性应该尽量低．

提取纹理特征的方法可以分为空间域方法和变换域方法 [1]，其中傅氏变换方法最为常用．傅氏变换建立了空间域信息与频域信息的对应，不同频率的能量谱和不同方向的能量谱反映了一定的图像特征 [2,3]．如果图像灰度变化很小或者较慢，则在低频处，能量谱值较大；如果图像灰度变化频繁或者较快，则在高频处，能量谱值较大；如果图像中有一条朝向 (与 X 轴夹角) 为 θ 的边缘，那么沿着与此边缘垂直的方向上，即 $\theta \pm \pi/2$ 的方向上，傅氏变换能量谱上也有明显的边缘 [2,4]．再者傅氏变换具有快速算法．Gabor 变换是纹理分析中最常用的提取纹理特征的方法之一 [5]．本质上讲，Gabor 变换是加窗傅氏变换，它在算法上也是基于傅氏变换．

利用傅氏变换作图像分析的典型做法要求计算滑动矩形窗的傅氏变换 [3,6−8]．即以图像中每一点为中心，取大小固定的矩形窗，分别计算各点矩形窗数据的傅氏变换或者是 Gabor 变换，然后利用各点矩形窗的变换数据，计算各种特征量．此时，矩形窗在图像平面域中以各像素点为中心水平或者垂直滑动，如图 1 所示．在已报道的文献中，在各个窗上的傅氏变换是分别计算的．本文提出了二维滑动矩形窗傅氏变换的快速递推算法，能大大降低滑动矩形窗傅氏变换和 Gabor 变换的计算复杂性．本文具体安排如下：第 2 节给出二维滑动矩形窗傅氏变换的快速递推算法的原理，第 3 节给出该快速递推算法复杂度分析和试验结果，第 4 节讨论该算法的扩展——二维滑动矩形窗 Gabor 变换的快速递推算法，第 5 节为全文总结。

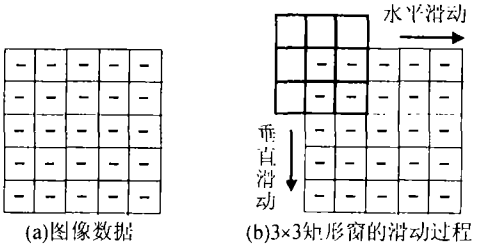

图 1 二维滑动矩形窗的滑动过程示意

[1] 2002-05-29 收到，2002-12-23 改回

2 算法原理

不失一般性，以下仅讨论水平方向滑动窗口的傅氏变换．考虑水平方向相邻两点矩形窗数据傅氏变换的关系．记图像数组为 A，数组尺寸为 $M\times N$．以像元 (m,n) 为中心的矩形邻域尺寸为 $M_w\times N_w(M_w, N_w$ 为奇数)，该邻域数组为 $D_{(m,n)}$, $M_1=(M_w-1)/2$, $N_1=(N_w-1)/2$．

滑动窗数据记为 $D_{(m,n)}=A(m-M_1:m+M_1,n-N_1:n+N_1)$, $D_{(m,n+1)}=A(m-M_1:m+M_1,n-N_1+1:n+N_1+1)$, $D_{(m+1,n)}=A(m-M_1+1:m+M_1+1,n-N_1:n+N_1)$．

计算上述滑动窗傅氏变换的传统方法是将这水平相邻或垂直相邻的两组数据看作是独立的数据，分别计算二者的傅氏变换．

观察水平相邻的窗口数据 $D_{(m,n)}$ 和 $D_{(m,n+1)}$，可以看出 $D_{(m,n)}$ 的第 2 列至第 N_w 列的数据与 $D_{(m,n+1)}$ 的 1 列至 N_w-1 列的数据完全相同，$D_{(m,n+1)}$ 的数据是 $D_{(m,n)}$ 的数据左移一列，再添加第 N_w 列新数据得到的，这实质就是滑动窗口的含义．利用 $D_{(m,n)}$ 和 $D_{(m,n+1)}$ 数据之间的这种滑动关系，可以直接从 $D_{(m,n)}$ 的傅氏变换递推出 $D_{(m,n+1)}$ 的傅氏变换，而无须重新计算 $D_{(m,n+1)}$ 的傅氏变换．

以下推导 $D_{(m,n)}$ 傅氏变换和 $D_{(m,n+1)}$ 傅氏变换的关系，记 $D_{(m,n)}$ 傅氏变换为 $\tilde{D}_{(m,n)}$, $D_{(m,n+1)}$ 傅氏变换为 $\tilde{D}_{(m,n+1)}$．

采用矢量表示形式，重写 $D_{(m,n)}$ 和 $D_{(m,n+1)}$．记 $D_{(m,n)}=[\boldsymbol{a}\ \ \boldsymbol{B}]$, $D_{(m,n+1)}=[\boldsymbol{B}\ \ \boldsymbol{c}]$，这里 $\boldsymbol{a}=D_{(m,n)}(:,1)$, $\boldsymbol{B}=D_{(m,n)}(:,2:N_w)=D_{(m,n+1)}(:,1:N_w-1)$, $\boldsymbol{c}=D_{(m,n+1)}(:,N_w)$．

$D_{(m,n+1)}$ 可重写为：$D_{(m,n+1)}=[\boldsymbol{B}\ \boldsymbol{c}]=[\boldsymbol{B}\ \boldsymbol{a}]-[\boldsymbol{0}\ \boldsymbol{a}]+[\boldsymbol{0}\ \boldsymbol{c}]$，这里 $\boldsymbol{0}$ 为 $(M_w-1)\times(N_w-1)$ 的零矩阵．由二维傅氏变换的定义可知：

$$
\begin{aligned}
\tilde{D}_{(m,n+1)} &= \sum_{m=0}^{M_w-1}\sum_{n=0}^{N_w-1} D_{(m,n+1)}e^{-j(2\pi/M_w)pm}e^{-j(2\pi/N_w)qn}\\
&= \sum_{m=0}^{M_w-1}\sum_{n=0}^{N_w-1}([\boldsymbol{B}\ \ \boldsymbol{a}]-[\boldsymbol{0}\ \ \boldsymbol{a}]+[\boldsymbol{0}\ \ \boldsymbol{c}])e^{-j(2\pi/M_w)pm}e^{-j(2\pi/N_w)qn}\\
&= \sum_{m=0}^{M_w-1}\sum_{n=0}^{N_w-1}[\boldsymbol{B}\ \ \boldsymbol{a}]e^{-j(2\pi/M_w)pm}e^{-j(2\pi/N_w)qn}\\
&\quad+\sum_{m=0}^{M_w-1}\sum_{n=0}^{N_w-1}[\boldsymbol{0}\ \ \boldsymbol{c}]e^{-j(2\pi/M_w)pm}e^{-j(2\pi/N_w)qn}\\
&\quad-\sum_{m=0}^{M_w-1}\sum_{n=0}^{N_w-1}[\boldsymbol{0}\ \ \boldsymbol{a}]e^{-j(2\pi/M_w)pm}e^{-j(2\pi/N_w)qn}
\end{aligned}
\tag{1}
$$

这里，$p=0,1,\cdots,M_w-1$, $q=0,1,\cdots,N_w-1$．

考虑 (1) 式中的第 1 项．矩阵 $[\boldsymbol{B}\ \ \boldsymbol{a}]$ 是由矩阵 $D_{(m,n)}$ 循环左移一列得到，利用傅氏变换平移性质可知，

$$
\mathrm{FFT}([\boldsymbol{B}\ \ \boldsymbol{a}])=\tilde{D}_{(m,n)}e^{j(2\pi/N_w)q} \tag{2}
$$

考虑 (1) 式中的第 2 项的傅氏变换,

$$\sum_{m=0}^{M_w-1}\sum_{n=0}^{N_w-1}[\mathbf{0}\ \ \boldsymbol{c}]e^{-j(2\pi/M_w)pm}e^{-j(2\pi/N_w)qn}=\sum_{m=0}^{M_w-1}e^{-j(2\pi/M_w)pm}\sum_{n=0}^{N_w-1}[\mathbf{0}\ \ \boldsymbol{c}]e^{-j(2\pi/N_w)qn}$$

$$=\sum_{m=0}^{M_w-1}e^{-j(2\pi/M_w)pm}\cdot\boldsymbol{c}(m)\cdot e^{-j(2\pi(N_w-1)/N_w)q}$$

$$=e^{j(2\pi/N_w)q}\cdot\sum_{m=0}^{M_w-1}e^{-j(2\pi/M_w)pm}\cdot\boldsymbol{c}(m)\tag{3}$$

同理可知, (1) 式的第 3 项的傅氏变换等于下式:

$$-e^{j(2\pi/N_w)q}\cdot\sum_{m=0}^{M_w-1}e^{-j(2\pi/M_w)pm}\cdot\boldsymbol{a}(m)\tag{4}$$

综合 (2), (3) 和 (4) 式可得出水平相邻的窗口数据 $D_{(m,n)}$ 和 $D_{(m,n+1)}$ 的傅氏变换存在以下递推关系:

$$\tilde{D}_{(m,n+1)}=\tilde{D}_{(m,n)}e^{j(2\pi/N_w)q}+e^{j(2\pi/N_w)q}\cdot\sum_{m=0}^{M_w-1}e^{-j(2\pi/M_w)pm}\cdot(\boldsymbol{c}(m)-\boldsymbol{a}(m))\tag{5}$$

同理, 可得出垂直相邻的窗口数据 $D_{(m,n)}$ 和 $D_{(m+1,n)}$ 的傅氏变换存在以下递推关系:

$$\tilde{D}_{(m+1,n)}=\tilde{D}_{(m,n)}e^{j(2\pi/M_w)p}+e^{j(2\pi/M_w)p}\cdot\sum_{n=0}^{N_w-1}e^{-j(2\pi/N_w)qn}\cdot(\boldsymbol{c}(n)-\boldsymbol{a}(n))\tag{6}$$

此时, $\boldsymbol{c}$ 和 $\boldsymbol{a}$ 为行矢量.

计算逆傅氏变换时, 相邻数据块逆傅氏变换的递推关系推导类似于以上过程, 其递推关系如下:

当计算一幅图像的滑动窗傅氏变换或逆傅氏变换时, 不需计算所有滑动窗口的傅氏变换, 只需计算第一个滑动窗口数据的傅氏变换, 即以图像元 $D_{(m,n)}$ 为中心的滑动窗口的傅氏变换或逆傅氏变换, 然后利用 (5) 式的水平递推关系式, 可以推导出同一行滑动窗口的傅氏变换; 利用 (6) 式的垂直递推关系式, 可以推导出同一列的滑动窗口傅氏变换; 这样, 通过计算一次傅氏变换, 就可利用递推关系 (5) 式和 (6) 式推导出整个图像滑动窗口的傅氏变换.

3 算法实现, 复杂度分析, 试验结果

本节给出计算滑动窗口傅氏变换的传统直接法和本文快速递推法的算法实现, 复杂度分析以及试验结果, 以进行性能比较.

3.1 算法实现

传统直接法直接计算各滑动窗口的傅氏变换. 其算法实现流程如图 2 所示.

快速递推法只直接计算第一块滑动窗口的傅氏变换, 然后利用递推关系递推计算出其它滑动窗口的傅氏变换, 其算法实现流程如图 3 所示.

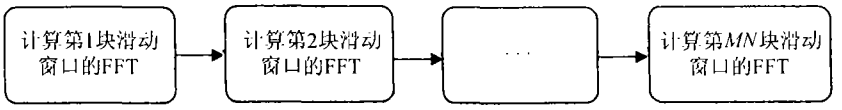

图 2　二维滑动矩形窗 FFT 直接法流程图

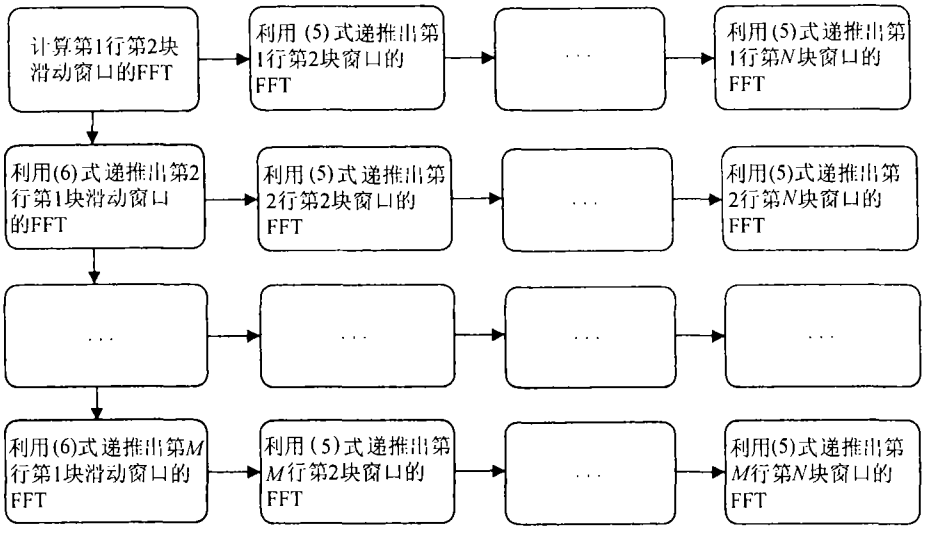

图 3　二维滑动矩形窗 FFT 递推法流程图

3.2 复杂度分析

如前定义，设图像尺寸为 $M \times N$，窗口尺寸为 $M_w \times N_w$。以下分析直接法和递推法的算法复杂度。

两种方法都需计算第 1 行第 1 列滑动窗口的 FFT，该计算复杂度为 $O(N_w^2 \cdot M_w + M_w^2 \cdot N_w)$，因此直接法的算法复杂度为：

$$O(\text{直接法}) = O((N_w^2 \cdot M_w + M_w^2 \cdot N_w) \cdot M \cdot N)$$

递推法水平递推一次的复杂度为 $O(M_w \cdot N_w + 2 \cdot M_w + M_w \cdot N_w)$，总的水平递推计算复杂度为 $O((2 \cdot M_w \cdot N_w + 2 \cdot M_w) \cdot M \cdot (N-1))$；递推法垂直递推一次的复杂度为 $O(M_w \cdot N_w + 2 \cdot N_w + M_w \cdot N_w)$，总的垂直递推计算复杂度为 $O((2 \cdot M_w \cdot N_w + 2 \cdot N_w) \cdot (M-1))$；因此递推法的算法复杂度为 $O(\text{迭代法}) = O[N_w^2 \cdot M_w + M_w^2 \cdot N_w + (2 \cdot M_w \cdot N_w + 2 \cdot M_w) \cdot M \cdot (N-1) + (2 \cdot M_w \cdot N_w + 2 \cdot N_w) \cdot (M-1)]$。

为了方便讨论，设 $M = N$，$M_w = N_w$，则两种方法的算法复杂度可分别写为

$$\begin{aligned} O(\text{直接法}) &= O((N_w^2 \cdot M_w + M_w^2 \cdot N_w) \cdot M \cdot N) \\ &= O(2 \cdot M_w^3 \cdot M^2) = O(2 \cdot M_w^3) + O(2 \cdot M_w^3 \cdot (M^2 - 1)) \end{aligned} \tag{7}$$

$$\begin{aligned} O(\text{迭代法}) &= O[N_w^2 \cdot M_w + M_w^2 \cdot N_w + (2 \cdot M_w \cdot N_w + 2 \cdot M_w) \cdot M \cdot (N-1) \\ &\quad + (2 \cdot M_w \cdot N_w + 2 \cdot N_w) \cdot (M-1)] \\ &= O(2 \cdot M_w^3) + O(2 \cdot M_w^2 + 2 \cdot M_w) \cdot (M^2 - 1)) \end{aligned} \tag{8}$$

当 $M_w << M$ 时，$O(\text{直接法}) \approx O(2 \cdot M_w^3 \cdot (M^2-1))$, $O(\text{迭代法}) \approx O(2 \cdot M_w^2 + 2 \cdot M_w) \cdot (M^2-1)$，可以得出此时随着窗口尺寸的增大，直接法复杂度增长速度为窗口尺寸的立方，而递推法复杂度增长速度为窗口尺寸的平方。有些情况下，并不满足 $M_w << M$ 的条件，但观察 (7) 式和 (8) 式，两种方法复杂度的差别仍然是窗口尺寸立方和平方的差别。

因此，本文提出的滑动窗口傅氏变换的快速递推算法性能优于直接计算法。一方面计算同一图像大小和同一窗口尺寸，递推法的计算时间远小于直接法的计算时间；另一方面当窗口尺寸增大或者图像尺寸增大时，递推法的增长速度远小于直接法的增长速度。

3.3 试验结果

在 Matlab 环境下，本文分别测试了递推法和直接法对同一组数据的计算时间。第一组试验测试图像尺寸固定 (128×128)，滑动窗口尺寸变化 (从 3×3 至 45×45) 时，两种算法的性能。第二组试验测试滑动窗口尺寸固定 (11×11)，图像尺寸变化 (从 32×32 至 256×256) 时，两种算法的性能。表 1 和表 2 分别是试验数据，图 4 和图 5 是试验数据对应的散点图。

需要说明的是，为保证算法比较的基础是一样的，除了使用 Matlab 的编程语言编写快速递推法外，还特别注意需用 Matlab 编程语言自己编写 fft2 的子程序。Matlab 本身提供了工具箱子程序 fft2，但该程序的主体部分是使用 C 语言编写的，并且该程序调用的一些相关子程序为动态连接库形式。因此如果直接调用 Matlab 自身的 fft2，将会造成两种算法比较的基础不一样；而对两种算法都采用 Matlab 中最为基本的编程指令 (即矩阵加法和乘法指令)，而非高级的工具箱函数 (例如 fft2 等)，就可以保证两种算法比较的基础一致。由于使用 Matlab 编程语言测试两种算法的性能，因此各算法的运行时间要远大于用 C 编程环境下的运行时间，但是两种算法运行时间之比在两种编程环境下是基本相同的。

表 1 图像尺寸固定 (128×128)，窗口尺寸变化时各算法在 Matlab 环境下的运行时间 (s)

窗口尺寸	3	9	15	21	27	33	39	45
递推法	9.18	9.99	12.52	15.87	18.68	25.76	30.43	37.96
直接法	18.51	23.67	38.94	56.03	78.16	116.82	151.59	202.23

表 2 窗口尺寸固定 (11×11)，图像尺寸变化时各算法在 Matlab 环境下的运行时间 (s)

图像尺寸	32	64	96	128	160	192	224	256
递推法	0.66	2.64	5.72	10.22	15.87	22.85	31.04	40.48
直接法	1.65	6.48	14.5	25.81	40.32	58.11	79.09	106.88

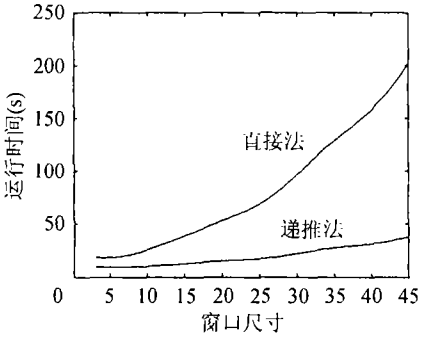

图 4 图像尺寸固定 (128×128)，窗口尺寸变化时各算法运行时间图

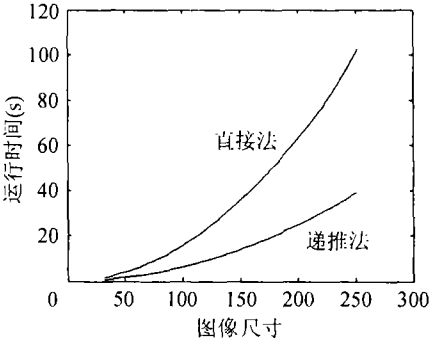

图 5 窗口尺寸固定 (11×11)，图像尺寸变化时各算法运行时间图

4 算法的扩展——二维滑动矩形窗 Gabor 变换的快速递推算法

Gabor 变换是频域加窗的傅氏变换，其原理是用一组带通滤波器作用于信号的傅氏变换，以提取图像傅氏变换的特定频带.

Gabor 变换可以表示为 $Y(p,q)=H(p,q)\cdot X(p,q)$，其中 $H(p,q)=\exp(-2\pi^2\sigma_p^2(p-p_0)-2\pi^2\sigma_q^2(q-q_0))$，$X(p.q)$ 是图像数据的傅氏变换，$Y(p,q)$ 为 Gabor 变换. 当计算二维滑动矩形窗的 Gabor 变换时，各滑动窗口所对应的 $H(p,q)$ 是不变的，变化的仅是滑动窗口的傅氏变换. 因此，可以把二维滑动矩形窗傅氏变换的快速递推算法扩展为二维滑动矩形窗 Gabor 变换的快速递推算法，算法流程如下：

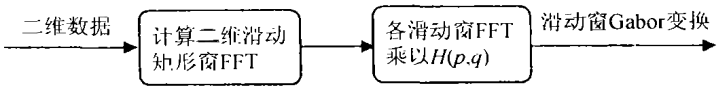

图 6　二维滑动矩形窗 Gabor 变换的快速递推算法流程图

5 结 束 语

本文提出了一种滑动矩形窗傅氏变换的快速递推算法，进一步将算法扩展为滑动矩形窗 Gabor 变换的快速递推算法. 算法复杂度分析以及试验结果表明，本文提出的快速递推法性能优于传统的直接计算法，大大降低了滑动窗傅氏变换的计算时间，尤其是当窗口尺寸和图像尺寸都比较大时. 本文结果可以极大地提高滑窗傅氏变换或 Gabor 变换的计算效率，这对图像分析，特别是对纹理图像分析具有重要意义.

参 考 文 献

[1] C. H. Chen, L. F. Pau, P. S. P. Wang, Texture Analysis, In The Handbook of Pattern Recognition and Computer Vision (2nd Edition), NY, World Scientific Publishing Co., 1998, 207-248.

[2] Zou Mouyan, Wang Dongfeng, Texture identification and image segmentation via Fourier transform, Proceedings for SPIE, 2001, Vol. 4550, 34-39.

[3] R. Azencott, J. P. Wang, L. Younes, Texture classification using windowed Fourier filters, IEEE Trans. on Patt. Anal. Mach. Intell., 1997, 19(2), 148-153.

[4] T. Randen, J. H. Husoy, Filtering for texture classification: a comparative study, IEEE Trans. on Patt. Anal. Mach. Intell., 1999, 21(4), 291-310.

[5] P. S. William, The automatic hierarchical decomposition of images into sub-images for use in image recognition and classification, [PHD Thesis], The University of Western Australia, Sep. 1999.

[6] C. M. Chen, H. H. Lu, K. C. Han, A textural approach based on Gabor functions for texture edge detection in ultrasound images, Ultrasound in Med. & Biol., 2001, 27(4), 515-534.

[7] D. Puig, M. A. GarCia, Determining optimal window size for texture feature extraction methods, IX Spanish Symposium on Pattern Recognition and Image Analysis, Castellon, Spain, May 2001, vol.2, 237-242.

[8] M. Koppen, J. R. Solar, P. Soille, Texture segmentation by biologically-inspired use of neural networks and mathematical morphology, Proc. of the International ICSC/IFAC Symposium on Neural Computation (NC'98), Vienna, Austria, September 23-25, 1998, 267-272.

A FAST RECURSIVE ALGORITHM FOR THE 2D SLIDING RECTANGULAR WINDOW FFT

Zhang Lifei Yang Hongbo Wang Dongfeng Zou Mouyan

(*Institute of Electronics, Chinese Academy of Sciences, Beijing* 100080, *China*)

Abstract In this paper, a fast recursive algorithm for the 2D sliding rectangular window FFT is proposed, based on the relationship of neighbor windows and the transformation property of the FFT. Further, the fast recursive algorithm is extended into 2D Gabor transformation of sliding rectangular windows. An analysis and a comparison on the computational complexity between the recursive algorithm and the traditional direct method are given. Both the theoretical analysis and the experimental result show that the new recursive algorithm can reduce the computation cost greatly, especially in the case of bigger image size or/and bigger window size. The proposed recursive algorithm can improve the computation efficiency of image analysis using the slide rectangular window FFT or Gabor transform.

Key words Image analysis, Texture analysis, FFT, Gabor transformation, Recursive algorithm

张丽飞： 女，1975 年生，博士生，研究方向为图像分割，图像复原和图像分析.
邹谋炎： 男，1941 年生，研究员，博士生导师，研究方向为信号处理，图像处理理论和算法，无线宽带通信技术.