Gabor 滤波器的快速实现

陈小光 [1] 封举富 [1]

摘 要 本文提出了 Gabor 滤波器的两种快速实现方法. 这两种方法首先把 Gabor 滤波器分解为多个不同方向上有着不同参数的一维高斯滤波器的组合, 然后通过递归的方法分别实现这些高斯滤波器, 从而实现 Gabor 滤波器. 实验和分析结果表明, 本文提出的 Gabor 滤波器的快速实现方法, 不论是计算复杂度还是计算精度, 都比基于卷积的标准实现方法有着更好的性能和效果.

关键词 Gabor 滤波, 高斯滤波器, 非正交分解, 递归滤波, 快速算法
中图分类号 TP391.41

Fast Gabor Filtering

CHEN Xiao-Guang[1] FENG Ju-Fu[1]

Abstract In this paper, two fast implementations of Gabor filter are proposed. The implementation strategy involves two steps: 1) decomposing Gabor filter into 1-D Gaussian filters along non-orthogonal axes with different variances; 2) recursively implementing these 1-D Gaussian filters. The non-orthogonal decomposition and recursive filtering ensure efficient performance. Experimental results showed that the proposed implementations outperform traditional convolution filtering with respect to computational speed and accuracy.

Key words Gabor filter, Gaussian filter, non-orthogonal decomposition, recursive filtering, fast algorithm

1 引言

在图像处理、模式识别以及计算机视觉等领域中, Gabor 滤波器得到了广泛的应用. Gabor 函数是唯一能够达到空域和频域联合测不准关系下界的函数[1], 用 Gabor 函数形成的二维 Gabor 滤波器具有在空间域和频率域同时取得最优局部化的特性, 因此能够很好地描述对应于空间频率 (尺度)、空间位置及方向选择性的局部结构信息. 20 世纪 90 年代以来, 人们对 Gabor 滤波器在纹理分割[1]、物体检测与识别[2]、图像增强[3] 以及图像特征提取[4,5] 等方面的应用做了很多研究. 常用的偶对称二维 Gabor 滤波器可表示为

$$h(x,y)=\frac{1}{2\pi\sigma_u\sigma_v}\exp\left\{-\frac{1}{2}\left(\frac{u^2}{\sigma_u^2}+\frac{v^2}{\sigma_v^2}\right)\right\}\cos(\omega u) \tag{1}$$

$$u=x\cos\theta+y\sin\theta,\quad v=-x\sin\theta+y\cos\theta \tag{2}$$

其中 θ 为 Gabor 滤波器的方向, σ_u 和 σ_v 分别为高斯包络在 u 轴和 v 轴上的标准差 (u 轴平行于 θ、v 垂直于 θ), ω 用于调制频率.

通常我们对图像进行滤波采用的实现方法是让图像与滤波器模板做卷积操作, 假如要用模板大小为 $N\times M$ 的滤波器 H 对图像 I 进行滤波, 则滤波结果 $\tilde{I}$ 为

$$\tilde{I}(x,y)=\sum_{n=-\lfloor\frac{N}{2}\rfloor}^{\lfloor\frac{N}{2}\rfloor}\sum_{m=-\lfloor\frac{M}{2}\rfloor}^{\lfloor\frac{M}{2}\rfloor}I(x+n,y+m)H(n,m) \tag{3}$$

对于 Gabor 滤波器, 具体实现时滤波器模板大小是由 σ_u 和 σ_v 决定的, 如 $N=[\alpha\,\sigma_u]$, $M=[\alpha\,\sigma_v]$, 其中 α 一般取值为 3、4 或 5, N 和 M 取奇数. 如果采用卷积实现滤波, 每个像素的计算量至少需要 NM 次乘法和 $NM-1$ 次加法, 共计 $2NM-1$ 次加乘法运算. 当 σ 取值为 5.0 时, 每像素的计算量超过 1000 次加乘法运算. 由此可见, 用卷积的方法实现 Gabor 滤波的计算复杂度是比较大的. 因此如果能设计一种 Gabor 滤波器的快速实现算法, 降低计算复杂度, 将会为 Gabor 滤波器的具体应用带来很大便利. 为此, 本文对 Gabor 滤波器的快速实现方法做了些初步研究.

本文下面几部分安排如下: 第 2 节给出了 Gabor 滤波器的正交和非正交分解形式, 把二维 Gabor 滤波器化简为简单的一维高斯滤波器的组合; 第 3 节给出了 Gabor 滤波器的快速实现方法, 并分析了计算复杂度; 第 4 节给出实验结果及分析讨论; 第 5

收稿日期 2006-1-23 收修改稿日期 2006-6-7
Received January 23, 2006; in revised form June 7, 2006
国家自然科学基金项目 (60575002), "973" 项目 (2004CB318000) 和 "新世纪优秀人才支持计划" 资助
Supported by National Natural Science Foundation of P. R. China (60575002), National Key Basic Research Project of P. R. China (2004CB318000), and the Program for New Century Excellent Talents in University
1. 北京大学信息科学技术学院智能科学系, 视觉与听觉信息处理国家重点实验室 北京 100871
1. State Key Laboratory of Machine Perception, Department of Machine Intelligence, Peking University, Beijing 100871
DOI: 10.1360/aas-007-0456

节是本文的结论.

2　Gabor 滤波器的可分解性质

为了尽量减少计算复杂度, 我们首先把 Gabor 滤波器分解为简单的一维滤波器的组合, 然后用计算量较小的递归滤波方法实现一维滤波. 下面给出两类分解形式, 一类是正交分解, 另一类是非正交分解; 递归实现在下一节给出.

2.1　正交分解

根据 (1), 我们可以直接把 Gabor 滤波器的表达式写为 u,v 变量分离的形式

$$h(x,y)=q(u)*g(v;\sigma_v) \tag{4}$$

其中 $q(u)$ 和 $g(v;\sigma_v)$ 分别为

$$q(u)=\frac{1}{\sqrt{2\pi}\sigma_u}\exp\left\{-\frac{u^2}{2\sigma_u^2}\right\}\cos(\omega u) \tag{5}$$

$$g(v;\sigma_v)=\frac{1}{\sqrt{2\pi}\sigma_v}\exp\left\{-\frac{v^2}{2\sigma_v^2}\right\} \tag{6}$$

当 $\sigma_u=\sigma_v=\sigma$ 时, 有 $u^2/\sigma_u^2+v^2/\sigma_v^2=(x^2+y^2)/\sigma^2$, 把此式以及 (2) 中的 u 代入 (1) 并展开 cos 项, 可得如下分解形式

$$h(x,y)=f_1(x)*f_2(y)-f_3(x)*f_4(y) \tag{7}$$

其中 f_1,f_2,f_3 和 f_4 分别为

$$\begin{aligned}
f_1(x)&=g(x;\sigma)\cos((\omega\cos\theta)x)\\
f_2(y)&=g(y;\sigma)\cos((\omega\sin\theta)y)\\
f_3(x)&=g(x;\sigma)\sin((\omega\cos\theta)x)\\
f_4(y)&=g(y;\sigma)\sin((\omega\sin\theta)y)
\end{aligned} \tag{8}$$

2.2　非正交分解

上节中的分解方法是把滤波器分解为沿着两个正交方向的一维滤波器的组合. 在分解形式 (4) 中 u,v 轴一般位于某个倾斜的角度上, 因此在实际实现时往往是在非整像素点上进行计算, 这就涉及到像素点的插值操作, 从而导致产生较大误差且计算量也增加许多. 另外, 当 $\sigma_u\neq\sigma_v$ 时, 指数项中包含 xy , 因此无法有形如 (8) 的分解, 此时只能采用二维图像与二维模板的卷积滤波实现方法 (见 (3)), 计算量将会相当可观.

Geusebroek 在文献 [6] 中给出了各向异性高斯滤波器的一种非正交分解方法. 各向异性高斯滤波器可以表示为

$$g(x,y)=\frac{1}{2\pi\sigma_u\sigma_v}\exp\left\{-\frac{1}{2}\left(\frac{u^2}{\sigma_u^2}+\frac{v^2}{\sigma_v^2}\right)\right\} \tag{9}$$

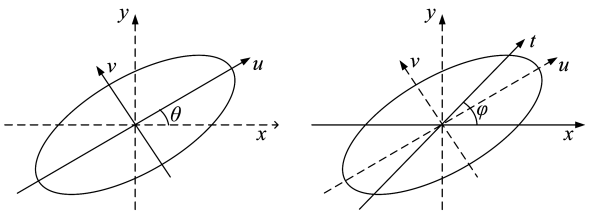


图 1　各向异性高斯滤波器的非正交分解[6]

Fig. 1　Non-orthogonal decomposition of anisotropic Gaussian filter[6]

$$u=x\cos\theta+y\sin\theta,\quad v=-x\sin\theta+y\cos\theta$$

如图 1 所示, 把各向异性的高斯滤波器分解为沿着 x 轴方向的一维高斯滤波器以及沿着直线 $t: y-x\tan\varphi=0$ 方向的一维高斯滤波器的组合,

$$g(x,y;\sigma_u,\sigma_v,\theta)=\frac{1}{\sqrt{2\pi}\sigma_x}\exp\left\{-\frac{x^2}{2\sigma_x^2}\right\}*\frac{1}{\sqrt{2\pi}\sigma_t}\exp\left\{-\frac{t^2}{2\sigma_t^2}\right\} \tag{10}$$

参数 $\sigma_x,\sigma_t,\varphi$ 的含义分别如下

$$\begin{aligned}
\sigma_x&=\frac{\sigma_u\sigma_v}{\sigma_v^2\cos^2\theta+\sigma_u^2\sin^2\theta}\\
\sigma_t&=\sqrt{\sigma_v^2\cos^2\theta+\sigma_u^2\sin^2\theta}\\
\tan\varphi&=\frac{\sigma_v^2\cos^2\theta+\sigma_u^2\sin^2\theta}{(\sigma_u^2-\sigma_v^2)\cos\theta\sin\theta}
\end{aligned} \tag{11}$$

把 (11) 中的结果以及坐标变换: $x\to x+t\cos\varphi$, $y\to t\sin\varphi$, 代入 Gabor 滤波器的表达式 (1) , 即得

$$h(x,y;\sigma_u,\sigma_v,\theta)=\frac{1}{\sqrt{2\pi}\sigma_x}\exp\left\{-\frac{x^2}{2\sigma_x^2}\right\}*\frac{1}{\sqrt{2\pi}\sigma_t}\exp\left\{-\frac{t^2}{2\sigma_t^2}\right\}*\cos(\omega_1x+\omega_2t) \tag{12}$$

其中 $\omega_1=\omega\cos\theta$, $\omega_2=\omega\cos(\theta-\varphi)$. 展开式 (12) 中的 cos 项并化简, 可得 Gabor 滤波器的 xt 分解形式

$$h(x,y)=h_1(x)*h_2(t)-h_3(x)*h_4(t) \tag{13}$$

其中 h_1,h_2,h_3 和 h_4 分别为

$$\begin{aligned}
h_1(x)&=g(x;\sigma_x)\cos(\omega_1x)\\
h_2(t)&=g(t;\sigma_t)\cos(\omega_2t)\\
h_3(x)&=g(x;\sigma_x)\sin(\omega_1x)\\
h_4(t)&=g(t;\sigma_t)\sin(\omega_2t)
\end{aligned} \tag{14}$$

2.3 $g(t)\cos(\omega t)$ 和 $g(t)\sin(\omega t)$ 的卷积性质

在上面两小节中, 我们给出了 Gabor 滤波器在不同参数条件下的一维分解形式 (见 (4), (7) 和 (13)). 可以注意到, Gabor 滤波器的分解表达式 (4) 中的 $q(\cdot)$、(7) 中的 $f_k(\cdot)$ 和 (13) 中的 $h_k(\cdot)$ 有着相同的形式, 都是一个高斯函数乘以 cos 或 sin 项. 虽然我们实现了 Gabor 滤波器的分解, 但是如果希望能快速实现 Gabor 滤波器, 我们必须能够快速实现此类一维滤波器. 在这里首先分析 $g(t)\cos(\omega t)$ 和 $g(t)\sin(\omega t)$ 的卷积性质, 然后在本文的第 3 节给出快速实现方法.

假设 $R(t)$ 是一个一维信号, 用 $g(t)\cos(\omega t)$ 对 $R(t)$ 进行卷积滤波, 根据卷积定义有

$$R(t)\otimes[g(t)\cos(\omega t)]=\int R(s)g(t-s)\cos(\omega(t-s))\,\mathrm{d}s,$$

展开上式右边的 cos 项可得,

$$R(t)\otimes[g(t)\cos(\omega t)]=\cos(\omega t)R_1(t)+\sin(\omega t)R_2(t) \tag{15}$$

其中 $R_1(t), R_2(t)$ 分别为

$$R_1(t)=g(t)\otimes(\cos(\omega t)R(t))$$
$$R_2(t)=g(t)\otimes(\sin(\omega t)R(t))$$

同理可得 $g(t)\sin(\omega t)$ 的卷积性质:

$$R(t)\otimes[g(t)\sin(\omega t)]=\sin(\omega t)R_1(t)-\cos(\omega t)R_2(t) \tag{16}$$

从 (15) 和 (16) 中可以看出, 一个信号与滤波器 $g(t)\cos(\omega t)$ 或 $g(t)\sin(\omega t)$ 的卷积可以通过比较简单的高斯滤波器来实现.

3 Gabor 滤波器的实现

3.1 $g(t)\cos(\omega t)$ 和 $g(t)\sin(\omega t)$ 的实现

在本文的第 2 节, 我们把 Gabor 滤波器分解为多个一维高斯滤波器的组合, 最后都涉及到用滤波器 $h_c(t)=g(t)\cos(\omega t)$ 或 $h_s(t)=g(t)\sin(\omega t)$ 对图像进行滤波. 在本小节中, 我们给出 $h_c(t)$ 的两种实现方法, 一种是基于卷积的实现, 另外一种是基于一维高斯滤波的递归实现[7]. $h_s(t)$ 的实现方法可以同理得到.

3.1.1 卷积实现

由于 $h_c(x)$ 的对称性, 假设其滤波器系数为 $\{w_{\lfloor N/2\rfloor},\cdots,w_1,w_0,w_1,\cdots,w_{\lfloor N/2\rfloor}\}$, 其中 N 为滤波器模板大小. 并假设 $I[x,y]$ 为输入图像, $O[x,y]$ 为滤波结果图像.

当滤波方向为 x 轴时, 卷积滤波的计算公式为式 (17). 当沿着倾斜角为 φ ($\pi/4\le\varphi<\pi/2$)、斜率为 $\mu=\tan\varphi$ 的直线 $t: y=\mu x$ 做滤波时, 计算公式为 (18).

注意在 (18) 中, 坐标 $y\pm j$ 是整数, 而 $x\pm j/\mu$ 有可能不是整数, 因此 $(x\pm j/\mu, y\pm j)$ 可能落在两个整像素点 $(\lfloor x\pm j/\mu\rfloor, y\pm j)$ 和 $(\lfloor x\pm j/\mu\rfloor+1, y\pm j)$ 之间, 其中 $\lfloor\cdot\rfloor$ 是向下取整符号. 此时可以通过这两个相邻的像素点线性插值得到“像素”点 $(x\pm j/\mu, y\pm j)$ 的值. 例如对于 $I[x-j/\mu, y-j]$, 记 $x_0=x-j/\mu$, $x_{left}=\lfloor x_0\rfloor$, $x_{right}=x_{left}+1$, 则有 (19).

当角度 $0<\varphi<\pi/4$ 或 $\varphi\ge\pi/2$ 时, 对图像进行转置或水平翻转, 即可归为上述情况.

$$O[x,y]=w_0I[x,y]+\sum_{i=1}^{\lfloor N/2\rfloor}w_i\,(I[x-i,y]+I[x+i,y]) \tag{17}$$

$$O[x,y]=w_0I[x,y]+\sum_{j=1}^{\lfloor N/2\rfloor}w_j\,(I[x-j/\mu,y-j]+I[x+j/\mu,y+j]) \tag{18}$$

$$I[x-j/\mu,y-j]=(x_{right}-x_0)I[x_{left},y-j]+(x_0-x_{left})I[x_{right},y-j] \tag{19}$$

3.1.2 递归实现

Young I T[7] 等人提出了一种递归实现一维高斯滤波的快速方法. 该方法在每个像素上只要求做 8 次乘法和 6 次加法, 计算量与高斯核的方差 σ^2 无关. 假设输入信号为 $IN[n]$, 滤波后信号为 $OUT[n]$, 则高斯滤波的递归实现过程可表示为式 (20). 其中常数 $\{a_0,a_1,a_2,a_3\}$ 是与高斯滤波器的方差 σ 有关的递归滤波系数, 这些系数的计算方法可查阅文献 [7].

结合递归高斯滤波方法以及 (15) 和 (16), 很容易得到 h_c 和 h_s 的递归实现方法. 例如, 沿着倾斜角为 φ ($\pi/4\le\varphi<\pi/2$)、斜率为 $\mu=\tan\varphi$ 的直线 $t: y=\mu x$ 做滤波 $\tilde{I}_\varphi(x,y)=I(x,y)\otimes[g(t)\cos(wt)]$, 可以按照 (21) 来计算得到, 其中的 $I[x+\mu y, y]$ 等值也需要通过相邻像素点插值得到.

3.2 Gabor 滤波器的几种实现方法

综合前面的结果, 我们给出 Gabor 滤波器的几种实现方法, 其中方法 3、4 为本文提出的快速实现方法.

方法 1. 二维卷积实现 (记为 *conv*22). 1) 计算 Gabor 滤波器模板; 2) 按照式 (3) 与输入图像做二维卷积.

$$\begin{cases} Temp[n] = a_0 IN[n] + a_1 Temp[n-1] + a_2 Temp[n-2] + a_3 Temp[n-3] \\ OUT[n] = a_0 Temp[n] + a_1 OUT[n+1] + a_2 OUT[n+2] + a_3 OUT[n+3] \end{cases} \tag{20}$$

$$\begin{cases} Z_1^f[x,y] = Z_1^f[t] = a_0 \cos(wt) I[x+\mu y, y] + a_1 Z_1^f[t-1] + a_2 Z_1^f[t-2] + a_3 Z_1^f[t-3] \\ Z_1^b[x,y] = Z_1^b[t] = a_0 Z_1^f[t] + a_1 Z_1^b[t+1] + a_2 Z_1^b[t+2] + a_3 Z_1^b[t+3] \\ Z_2^f[x,y] = Z_2^f[t] = a_0 \cos(wt) I[x+\mu y, y] + a_1 Z_2^f[t-1] + a_2 Z_2^f[t-2] + a_3 Z_2^f[t-3] \\ Z_2^b[x,y] = Z_2^b[t] = a_0 Z_2^f[t] + a_1 Z_2^b[t+1] + a_2 Z_2^b[t+2] + a_3 Z_2^b[t+3] \\ \tilde{I}_\varphi[x,y] = \tilde{I}_\varphi[t] = \cos(wt) Z_1^b[t] + \sin(wt) Z_2^b[t] \end{cases} \tag{21}$$

方法 2. 在 uv 方向上一维卷积 (记为 $uvconv$). 1) 把 Gabor 滤波器分解为 (4) 中形式; 2) 让输入图像在 u 轴方向 (即 θ 角度方向) 上用 $q(u)$ 做滤波, 滤波方法采用 3.1 节中的卷积实现方法; 3) 对第 2) 步的结果在 v 轴方向 (即 $\theta+\pi/2$ 角度方向) 用一维高斯滤波器 $g(v;\sigma_v)$ 做滤波, 滤波实现方法采用卷积实现.

方法 3. 在 uv 方向上递归实现 (记为 $uvrec$). 1) 把 Gabor 滤波器分解为式 (4) 中形式; 2) 让输入图像在 u 轴方向 (即 θ 角度方向) 上用 $h_5(u)$ 做滤波, 滤波方法采用 3.1 节中 (21) 的递归实现方法; 3) 对第 2) 步的结果在 v 轴方向 (即 $\theta+\pi/2$ 角度方向) 用一维高斯滤波器 $g(v;\sigma_v)$ 做滤波, 滤波方法采用 (20) 的递归实现方法.

方法 4. 在 xt 方向上递归实现 (记为 $xtrec$). 1) 按照 2.2 节中的方法, 把 Gabor 滤波器分解为 (13) 的形式, 计算相应参数; 2) 按照 3.1 节中的方法, 对输入图像 I 用递归方法实现滤波, 得到 $I_1 = I \otimes h_1$ 和 $I_3 = I \otimes h_3$, 然后再用同样的递归方法实现滤波, 得到 $I_2 = I_1 \otimes h_2$ 和 $I_4 = I_3 \otimes h_4$; 3) 计算 $I_2 - I_4$ 得到最终结果.

3.3 计算复杂度分析

表 1 给出了上述四种实现方法在计算复杂度方面的比较, 其中 $W \times H$ 为图像大小, $N \times M$ 为采用卷积方法时滤波器模板大小. 由于 $g(t)$, $g(t)\cos(\omega t)$ 和 $g(t)\sin(\omega t)$ 都是对称的, 因此用卷积实现时可以利用对称性减少乘法次数 (见 (17), (18)). 当沿着某个倾斜的方向进行滤波时, 不位于整像素点上的像素值需要通过插值来得到, 因此增加了两次乘法和一次加法 (见 (19)). 在方法 $uvrec$ 中, 图像用 $q(u)$ 通过递归的方法进行滤波 (见 3.1 节), 每个像素需要做乘法 $2\times10+2$ 次及加法 $2\times7+1$ 次; 再与 $g(v)$ 通过递归方法进行滤波, 每个像素需要做乘法 9 次及加法 7 次. 在方法 $xtrec$ 中, 实现 $h_1(x)$ 和 $h_3(x)$, 每个像素共需要 2×18 次乘法及 2×7 次加法; 实现 $h_2(t)$ 和 $h_4(t)$, 每个像素共需要 2×22 次乘法及 2×8 次 加法; 完成整个过程每个像素共需要 80 次乘法及 31 次加法. 在上述这几种实现方法中, 由于方法 $uvrec$ 和 $xtrec$ 采用递归的滤波方法, 因此每个像素的计算量都是常数, 总的计算量只与图像大小有关; 而方法 $conv22$ 和 $uvconv$ 由于采用的是通常的卷积滤波, 计算量与滤波器模板大小有关, 也就是与 Gabor 滤波器中的参数 σ_u 和 σ_v 有关且成正比, 因此总的计算量会随着 σ_u 和 σ_v 的增大而快速增加.

表 1　Gabor 滤波不同实现方法的每像素计算量比较
Table 1　Computational complexity per pixel of various Gabor filter implementations

	加法次数	乘法次数
$conv22$	NM	$NM-1$
$uvconv$	$2(\lfloor N/2\rfloor + \lfloor M/2\rfloor + 2)$	$2(N+M-2)$
$uvrec$	31	22
$xtrec$	80	31
FFT	$\log(WH)$	$\log(WH)$

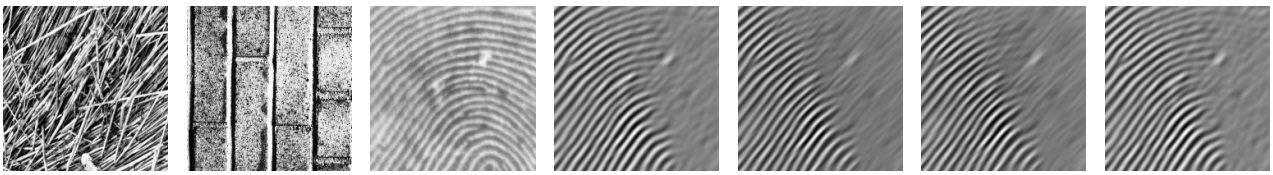
4　实验结果

为验证本文提出的方法的效果, 我们在 CPU 为 P4 1.7 GHz、内存 256M 的 PC 机上进行了性能对比实验, 编程语言为 VC6.0. 实验分为两部分, 第一部分是测试各种实现方法在计算时间上的差异, 第二部分是测试不同实现方法的计算误差的差异.

首先在 512×512 大小的图像上进行了计算时间的比较, 表 2 给出了实验结果, 表中数据是根据各方向上的计算时间的平均值得到的. 从实验结果可以看出, 本文提出的方法 $uvrec$ 和 $xtrec$ 在计算时间上确实大大少于方法 $conv22$ 和 $uvconv$, 而且基本上都是常数时间, 这与 3.3 节中的分析是一致的. 方法 $xtrec$ 与方法 $uvrec$ 由于分解形式不同 (见 (5),(13)), 使得主要计算步骤的时间代价基本上是前者为后者的两倍 (见表 1), 但由于具体编程实现时, 方法 $uvrec$ 需要在两个倾斜方向上进行滤波, 需要更多的时间及空间代价, 因而导致了这两种方法在完成整个滤波过程时所需要的时间差别不大.

表 2 Gabor 滤波不同实现方法的计算时间及误差比较
Table 2 Performance of various Gabor filter implementations

		时间 ($msec$)				误差 (ϵ)		
σ_u	σ_v	$conv22$	$uvconv$	$uvrec$	$xtrec$	$uvconv$	$uvrec$	$xtrec$
3.0	4.0	328	172	93	110	4.37	7.20	2.84
3.0	5.0	391	172	93	109	4.67	7.72	3.49
3.0	7.0	547	171	94	126	5.85	7.83	5.17
4.0	4.0	484	188	110	110	3.16	5.04	1.08
4.0	6.0	687	219	93	109	6.24	9.90	4.31
4.0	8.0	875	234	110	109	6.84	9.48	5.08
5.0	6.0	782	203	94	94	7.75	9.00	2.71
5.0	8.0	1047	265	109	110	8.20	9.65	3.50
5.0	10.0	1391	250	93	109	8.51	9.94	4.77
6.0	7.0	1063	234	94	94	8.50	9.29	1.94
6.0	8.0	1234	266	109	108	8.75	9.55	2.10
6.0	10.0	1625	282	93	94	8.78	9.66	2.84



(a) (b) (c) (c_1) (c_2) (c_3) (c_4)

图 2 部分测试图像及滤波结果图像 ((a)∼(c) 为部分测试纹理图像; (c_1) ∼ (c_4) 为对指纹图像 (c) 采用不同实现方法 $conv2$, $uvconv$, $uvrec$ 和 $xtrec$ 得到的 Gabor 滤波结果)
Fig. 2 (a) ∼ (c): some texture images for testing; (c_1) ∼ (c_4): Gabor filtering results of (c) using various implementations, $conv2$, $uvconv$, $uvrec$, and $xtrec$, respectively.

从表 2 中可以看出, 平均情况下方法 $xtrec$ 比方法 $uvrec$ 多 12 毫秒左右.

考虑到 Gabor 滤波器的具体应用, 我们选取了 Brodatz 纹理集[8] 中的 13 幅纹理图像以及 FVC2000[9] 给出的指纹图像数据 DB1-B 中的 10 幅指纹图像作为测试图像集 (记 $IS = \{I_k, k = 1, \cdots, 23\}$). 图 2 给出了实验中采用的部分纹理图像以及部分实验结果图像. 另外, 我们把 Gabor 滤波器的方向值量化为 18 个方向 (记 $DS = \{\theta_i = i * \pi/18, i = 0, \cdots, 17\}$). 误差用平均像素值差来度量, 并且以方法 $conv22$ 的结果作为标准, 然后计算其它方法得到的结果相对与它的误差. 另外, 由于 Gabor 滤波器可以位于不同方向上, 我们取所有测试图像在各个不同方向上的误差的平均值进行比较. 其计算公式为

$$\xi_m(\sigma_u, \sigma_v; \theta; I_k) = \frac{1}{NM} \sum_{x,y} |\tilde{I}_m(x,\ y) - \tilde{I}_{conv22}(x, y)|$$

$$\epsilon_m(\sigma_u, \sigma_v) = \frac{1}{18 \times 23} \sum_{\substack{\theta_i \in DS \\ I_k \in IS}} \xi_m(\sigma_u, \sigma_v; \theta_i; I_k)$$

这里 I_k 表示大小为 $N \times M$ 的测试图像, $\tilde{I}_{conv22}$ 表示用方法 $conv22$ 得到的滤波结果, $\tilde{I}_m$ 表示用方法 m 得到的滤波结果, Gabor 滤波器的参数为 $\sigma_u, \sigma_v, \theta, \omega$.

从表 2 中可以看出, 在相同参数条件下, 方法 $xtrec$ 的误差最小, $uvconv$ 其次, $uvrec$ 的误差最大. 在这三种方法中, $xtrec$ 的误差要比其它两种方法小许多, 这是因为方法 $uvrec$ 和 $uvconv$ 需要在两个倾斜的方向 (u 轴和 v 轴方向) 上进行插值计算, 引入了更多因插值带来的误差, 而方法 $xtrec$ 只需要在 t 轴方向插值计算. 与 $uvconv$ 相比较, 由于 $uvrec$ 采用了递归高斯滤波, 而高斯滤波的递归实现只是真实滤波的一种近似[6,7], 存在着一定的近似误差, 因此方法 $uvrec$ 的误差又稍大一些.

5 结论

本文对 Gabor 滤波器的快速实现方法做了初步研究, 提出了两种快速实现方法. 主要思想是, 首先把 Gabor 滤波器分解为不同方向 (正交的或不正交的) 上的多个一维滤波器的组合, 这些一维滤波器是高斯滤波器 $g(t; \sigma_t)$ 以及形如 $g(t; \sigma_t)\sin(\omega t)$ 或 $g(t; \sigma_t)\cos(\omega t)$ 的一类滤波器, 然后用递归的方法快速实现它们与图像的滤波. 本文提出的实现方法的

另一优点就是, 整个滤波过程的计算量只与图像的大小有关, 与 Gabor 滤波器的参数 σ_u 和 σ_v 无关. 实验结果表明, 本文提出的 Gabor 滤波器的快速实现方法与其它方法相比, 降低了时间代价, 减少了计算误差, 提高了滤波结果的精度, 因此是一种十分可行的实现方法.

References

1 Turner M. Texture discrimination by gabor functions. *Biological Cybernetics*, 1986, **55**: 71∼82

2 Jain A, Ratah N K, Lakshmanan S. Object detection using gaborfilters. *Pattern Recognition*, 1997, **30**(2): 295∼309

3 Hong L, Wan Y, Jain A. Fingerprint image enhancement: algorithm and performance evaluation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 1998, **20**(8): 777∼789

4 Zhu Y, Tan T, Wang Y. Biometric personal identification based on iris patterns. In: Proceedings of 15th International Conference on Pattern Recognition. Barcelona, Spain, 2000. 801∼804

5 Liu C, Wechsler H. Gabor feature based classification using the enhanced fisher linear discriminant model for face recognition. *IEEE Transactions on Image Processing*, 2002, **11**(2): 467∼476

6 Geusebroek J M, Smeulders A W M, Weijer J. Fast anisotropic gauss filtering. *IEEE Transactions on Image Processing*, 2003, **12**(88): 938∼943

7 Young I T, Vliet L J. Recursive implementation of the Gaussian filter. *Signal Processing*, 1995, **44**: 139∼151

8 Brodatz Texure Images[Online], available: http://sipi.usc.edu/services/database/database.cgi?volume=textures

9 Fingerprint Verification Competition (FVC2000)[Online], available: http://bias.csr.unibo.it/fvc2000/

陈小光 北京大学信息科学技术学院博士研究生, 研究领域为图像处理和生物特征识别. 本文通信作者.
E-mail: chenxg@cis.pku.edu.cn.
(**CHEN Xiao-Guang** Ph. D. candinate in School of Electronics Engineering and Computer Science at Peking University. His research interest covers image processing and biometrics. Corresponding author of this paper.)

封举富 理学博士, 北京大学信息科学技术学院智能科学系、视觉与听觉信息处理国家重点实验室教授, 博士生导师. 研究领域为图像处理和模式识别、生物特征识别等.
E-mail: fjf@cis.pku.edu.cn
(**FENG Ju-Fu** Received his B. Sc. degree in 1989 and Ph. D. degree in 1997 both in mathematics at Peking University. Since 1992, he has been with the Department of Machine Intelligence and State Key Laboratory of Machine Perception at Peking University. He is currently a professor in the Center for Information Science and State Key Lab of Machine Perception, School of Electronics Engineering and Computer Science, Peking University. His research interest covers image processing, pattern recognition, and biometrics.)