一种高效执行并行图像旋转的新方法

董育宁
(南京邮电学院信息工程系,江苏南京 210003)

摘 要: 本文提出了一种在分布式并行机上有效地执行二维图像旋转的方法. 算法的开发是在高级图像抽象的框架下进行的. 文中阐述了利用在二维可分旋转技术的基础上新开发的高级抽象表达式,可以简捷地表示并行图像旋转,并可在分布式存贮并行体系结构上有效地执行. 本文还对二维可分运算进行了误差分析,并提出了减小误差的补偿方案.

关键词: 并行图像旋转; 图像处理抽象; 可分旋转
中图分类号: TP391 **文献标识码**: A **文章编号**: 0372-2112 (2001) 12-1671-05

A New Method for Efficiently Implementing Parallel Image Rotations

DONG Yu-ning
(*Dept. of Information Engineering, Nanjing University of Posts and Telecommunications, Nanjing, Jiangsu* 210003, *China*)

Abstract: A method for efficiently implementing 2D image rotations on distributed memory parallel machines, is proposed in this paper. The method has been developed in the context of developing the algorithms in terms of high level abstractions. With newly developed notations based on a 2-pass rotation technique, we show that the parallel image rotation can be expressed concisely and performed effectively on distributed memory parallel architectures. Error analysis has been done for the 2-pass operations, and compensation schemes suggested to reduce noises.

Key words: parallel image rotations; image processing abstractions; 2-pass rotations

1 引言

近年来,在图像处理应用中,越来越多地使用了并行计算机体系结构[1,2]. 因为对于大多数图像处理研究人员来说,进行并行编程难度较大,因此需要高级编程接口,它可以使用户免受并行系统日益增加的复杂度的影响. 但是,如果没有适当的实现方法,它会在一定程度上降低系统的效率. 基于这些考虑,已出现了许多面向高级并行图像处理的编程抽象或语言,它们能够使程序员将精力关注于图像处理方面的问题[3~6].

与其他已发表的方法相比,Crookes 等人的图像处理抽象(CIPA)[6]有一个明显的优点,即既有高级语言的灵活表达方式,又可方便、高效地并行执行. 它的抽象表达式能对许多流行的算法进行简洁地编程. 英国贝尔法斯特女皇大学和乌尔斯特大学在 CIPA 的并行实现方面做了开创性的研究.

尽管 Crookes 等人的图像处理抽象很有希望成为一个并行图像处理方面有效的软件工具,但对某些图像处理运算,包括图像旋转,如何在分布式并行机上有效地执行,这个问题仍然没有解决.

分布式存储系统对于同时发生的大量的本地运算是很有效的,但是如果处理器间需要大量的通信(如一个处理器需要存取不在本地存储器中的数据),则系统的连接网络就可能过载. 因此系统的整体性能将会急剧下降,因为通信占用的处理器指令周期是相当多的. 所以,在分布式多处理器系统上编程图像处理算法的一个目标,便是尽可能地减少处理器间的通信. 标准的图像旋转通常被看作是一个全局运算. 尽管在概念上它们是邻域运算(只是该邻域运算的模板是变化的,并且在行与列方向都是无约束的,详见下节),但通常不这么认为. 这就有可能会影响它们在分布式并行机上的执行效率.

本文将讨论在中等或粗颗粒的 MIMD 分布式多处理器系统上,(被看作为全局运算的)图像旋转的有效计算问题. 在这样一个并行机上,每个处理器拥有一个本地存储器,其中至少存有若干整行或整列的图像数据,并且每个处理器可独立地处理本地的子图像. 英国的贝尔法斯特女皇大学和乌尔斯特大学已有这样一个系统——EPIC(可扩展并行图像协处理器)[7,8]. 本文的标记方法按照 CIPA 的模式.

2 问题的陈述

本节将指出并行图像旋转的计算效率问题.

设 $I_h(i,j)$, $0 \le i < N_i$ 且 $0 \le j < N_j$, 表示由离散输入图像 $I_g(p,q)$, $0 \le p < N_p$ 且 $0 \le q < N_q$, 旋转而产生的离散输出图像,如图 1 所示. 绕原点旋转输入图像的操作,可由反向地址映射(RAC)求得[10]:

$$\begin{cases} p = i\cos\theta - j\sin\theta & (1a) \\ q = i\sin\theta + j\cos\theta & (1b) \end{cases}$$

收稿日期:2000-04-06;修回日期:2001-10-17

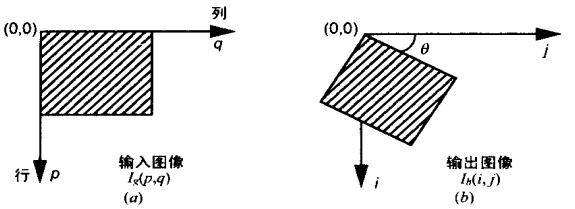

图 1 图像旋转 θ 角度. (a) 输入图像; (b) 输出图像

或由正向地址映射 (FAC) 求得:

$$\begin{cases} i = p\cos\theta + q\sin\theta & (2a) \\ j = -p\sin\theta + q\cos\theta & (2b) \end{cases}$$

其中 θ 是将输入图像相对于水平轴顺时针方向旋转的角度.

在 CIPA 中,图像绕原点旋转的 RAC (式 (1)) 可用图像模板操作表示 (输入图像与模板的卷积)[6],

$$I_h = \text{Convolve}\ (\text{Template}, I_g) \quad (3a)$$

其中模板的定义如下 (详见文[6]),

$$\begin{aligned} \text{Rotate at } i,j &= [\,< p - i, q - j, 1 >\,] \\ &= [\,< i(\cos\theta - 1) - j\sin\theta, i\sin\theta + j(\cos\theta - 1), 1 >\,] \end{aligned} \quad (3b)$$

由于对位于 (i,j) 处的某输出像素,计算出的输入像素位置 (p,q) 可在图像中任一点 (由旋转角度 θ 决定),这不是一个局部运算,将给并行执行带来困难.

下一节将提出一个实用且较好的解决方法,它利用了二维可分旋转技术.

3 适用于可分旋转的高级抽象

可分旋转算法主要是将二维图像的旋转分解为两个分开的、在行和列方向上的一维数据重洗. 按照 Catmull 和 Smith 的理论[9],可分旋转可由下述方法实现.

分解 1 ——行方向数据重排序 (以下简称行重洗)

按 FAC (式 2(b)) 或 RAC 运算,将 $I_g(p,q)$ 的每一行映射到一个中间图像 $I_1(p,j)$ 相应的行,如下所示.

由式 (2b) 可得,

$$q = \frac{p\sin\theta + j}{\cos\theta} = j\sec\theta + p\tan\theta \quad (4)$$

若定义模板 T_1 如下,

$$\begin{aligned} T_1 \text{ at } p,j &= [\,< 0, q - j, 1 >\,] \quad (\text{由式 (4)}) \\ &= [\,< 0, (\sec\theta - 1) * j + \tan\theta * p, 1 >\,] \end{aligned}$$

则由 RAC 计算得到的行重洗,可用下式表示,

$$I_1 = \text{Convolve}[\,T_1, I_g\,]$$

分解 2 ——列方向数据重排序 (以下简称列重洗)

在第二遍数据重洗中,$I_1(p,j)$ 的每一列经过处理,得到 $I_h(i,j)$ 中相应的列. 与第一遍重洗相似,可由 FAC 或 RAC 计算得到.

对于 FAC 计算,将式 (4) 代入式 (2a),得到

$$\begin{aligned} i &= p\cos\theta + \sin\theta\frac{p\sin\theta + j}{\cos\theta} \\ &= \frac{p(\sin^2\theta + \cos^2\theta) + j\sin\theta}{\cos\theta} = \frac{p + j\sin\theta}{\cos\theta} \\ &= p\sec\theta + j\tan\theta \end{aligned} \quad (5)$$

式 (1a) 可直接用于 RAC 计算. 若定义模板 T_2 如下,

$$\begin{aligned} T_2 \text{ at } i,j &= [\,< p - i, 0, 1 >\,] \quad (\text{由式 (1a)}) \\ &= [\,< (\cos\theta - 1) * i - \sin\theta * j, 0, 1 >\,] \end{aligned}$$

由 RAC 计算得到的列重洗,可用下式表示

$$I_h = \text{Convolve}[\,T_2, I_1\,]$$

注意 显而易见,当旋转角度为 90 时,上述分步计算不能进行. 这便是瓶颈问题, Wolberg [11, Ch. 7], Catmull 和 Smith[9] 对此问题已做了深入的讨论. 为了避免这个问题,对于较大角度的旋转 ($|\theta| > 45°$),可通过几个分开的步骤完成. 要得到最佳结果,应尽可能小地移动像素. 因此,旋转 75 可通过先旋转 90°,再旋转 - 15 来完成 (应注意到,对于 90 度的旋转,无任何信息损失). 图像旋转任一角度 θ, $0 \le \theta \le 360°$,可用下式中的一个来完成:

$$\theta = \begin{cases} \theta' \\ \theta' \pm 90° & -45° \le \theta' \le 45° \\ \theta' \pm 180° \end{cases} \quad (6)$$

用于 $-45° \le \theta \le 45°$ 的基本算子

若将行方向数据重洗算子定义为

$$\text{RowPass at } p,j = \text{Convolve } T_1$$

列方向数据重洗算子定义为

$$\text{ColPass at } i,j = \text{Convolve } T_2$$

复合算子定义为

$$\text{Rotate45} = \text{RowPass}^{\circ}\text{ColPass}$$

则对于 $|\theta| \le 45°$,整个运算为 (见图 2)

$$\text{Result} = \text{Rotate45}\ (\text{Input\_image})$$

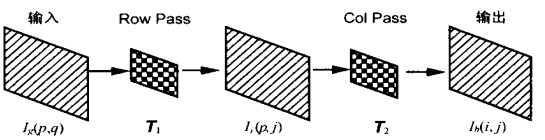

图 2 可分图像旋转算法

$0 \le \theta \le 360°$ 的算子

对于特殊角度的旋转,定义另外的模板和算子.

(1) $\pm 90°$ 旋转

$$T_{90} \text{ at } i,j = [\,< -j - i, i - j, 1 >\,]$$

$$T_{-90} \text{ at } i,j = [\,< j - i, -i - j, 1 >\,]$$

$$\text{Rotate90} = \text{Convolve}[\,T_{90}, \text{Image}\,]$$

$$\text{Rotate}M90 = \text{Convolve}[\,T_{-90}, \text{Image}\,]$$

(2) 180 旋转 ($\pm 180°$ 旋转是一样的)

$$T_{180} \text{ at } i,j = [\,< -2i, -2j, 1 >\,]$$

$$\text{Rotate180} = \text{Convolve}[\,T_{180}, \text{Image}\,]$$

对于任一角度的旋转, $0 \le \theta \le 360°$,可由式 (6) 中的一个式子,并结合上面定义的 Rotate45, Rotate90, Rotate$M$90, 和 Rotate180 来计算.

在二维可分旋转过程中,需要将中间图像进行一次全局性的转置运算 (从行分布模式变为列分布模式). 考虑到可用内置函数来执行转置运算,而该内置函数是用优化的低层代码编制的,所以有理由相信这种可分旋转方法比传统的旋转方法性能要好. 传统方法在计算过程中需要的通信数量是不

可预测的.

4　对可分旋转方法的误差分析

可分旋转的确有助于使得计算数据本地化,但是,即使瓶颈问题能够避免,它能否产生与传统方法一样的结果呢?下面将对此进行讨论,并考虑到量化误差.

4.1　FAC 计算

(1) 分解 1——按式(2b)进行行重洗

将式(2b)中的整数 j 用实数 u 代替,得

$$u = -p\sin\theta + q\cos\theta \tag{7}$$

当 u 四舍五入为整数 j 时,即

$$j = \mathrm{Round}(u) = u + \delta \tag{8}$$

行方向上就引入了舍入误差 δ, $-0.5 \le \delta < 0.5$,一般舍入误差 δ 会被带入第二次分解. 这样,相对于式(1)的直接计算,可分算法的计算误差就增大了.

(2) 分解 2——按式(5)进行列重洗

同样的,用实数 v 替换 i,得

$$v = \frac{p + j\sin\theta}{\cos\theta} = p\sec\theta + j\tan\theta \tag{9}$$

或由式(8)得

$$v = p\sec\theta + (u + \delta)\tan\theta = p\sec\theta + u\tan\theta + \delta\tan\theta \tag{10}$$

式(10)中的第三项 $\delta\tan\theta$ 是由第一次分解的误差扩散形成的. 若将旋转角度 θ 限制在 ±45 度以内,则误差就 $\le|\delta|$.

4.2　RAC 计算

(1) 分解 1——按式(4)进行行重洗

用实数 x 代替 q,得

$$x = j\sec\theta + p\tan\theta \tag{11}$$

其中 j, p 是整数. 然而 p 本应是由式(1a)经第二次分解计算而得的值,而这一般会是一个实数. 将这个值四舍五入为整数,就会产生舍入误差 e, $-0.5 \le e < 0.5$. 因此在 x 四舍五入成为整数 q 之前,受到了误差项 $e\tan\theta$ 的影响. 同样,如果将旋转角度限制在 ±45 度以内,则误差就 $\le|e|$. 后面将看到,与传统的旋转方法不同,这个误差将会使旋转后的图像质量劣化.

(2) 分解 2——按式(1a)进行列重洗

用实数 y 代替 p,得

$$y = i\cos\theta - j\sin\theta = p + e,\ -0.5 \le e < 0.5 \tag{12}$$

其中, p 是 y 四舍五入后的值, e 表示列方向上的舍入误差. 系数 i 和 j 是真实值,本式中没有从第一次分解带来的误差扩散.

很明显,由于在中间步骤引入了误差,在数学上看来很完美的可分解过程,会产生比传统方法质量差一些的图像. 因此需要特别的考虑来消除这些误差.

5　误差补偿

由于图像网格的离散特性,RAC 计算通常会产生在输入图像的已知像素值间的非线性地址结果,如图 3 所示. 为了得到最佳的结果,可以用某些重抽样方法,来估计输出的像素值. 下面就介绍两种流行的重抽样方法,以消除可分旋转法引入的额外误差.

最近邻居法:这种方法既可用于 FAC 计算,又可用于 RAC 计算. 如上节所述,由于第一次分解引入的舍入误差,以及它在第二次分解时发生的扩散,用模板 T_1 和 T_2 直接执行可分旋转,不一定能得到最近邻居输出像素值. 然而,如果我们保存初始的列坐标值 u(式(7))及第一次分解过程中的像素值,则误差可在第二次分解时得到补偿,但这需要额外的存储空间. 这种方法主要是用保存的 u 值来代替式(5)中的 j 值来实现的. 这样就能生成与标准旋转方法一样的最近邻居输出图像.

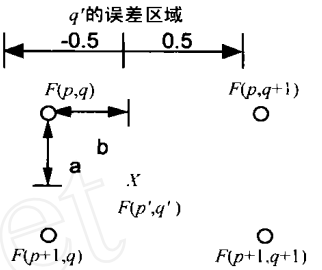

图 3　双线性内插和行方向上可能的误差区域

双线性内插:这种重抽样方法通常与 RAC 计算结合使用. 双线性内插就是对图像的每一行进行线性内插,再对得到的结果在列方向上进行线性内插,图 3 例示了这个过程. 其中 F 代表像素的灰度值. 不在整数点的地址 (p', q'),其灰度值用下式近似,

$$F(p', q') = (1-a)[(1-b)F(p,q) + bF(p,q+1)] + a[(1-b)F(p+1,q) + bF(p+1,q+1)] \tag{13}$$

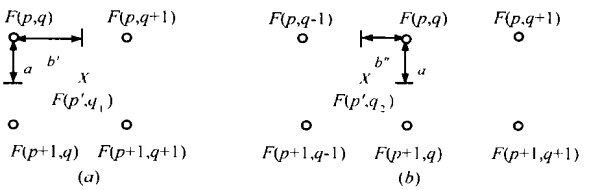

图 4　q 的错误计算

若定义如下两个模板,

$$T_3 \text{ at } p, j = [<0, q-j, 1-b>, <0, q+1-j, b>] \tag{14}$$

$$T_4 \text{ at } i, j = [<p-i, 0, 1-a>, <p+1-i, 0, a>] \tag{15}$$

那么在行和列分解中,用 T_3 和 T_4 分别代替 T_1 和 T_2,就可在可分旋转中粗略地实现双线性内插.

但是,由于在中间步骤引入了误差,它们与式(13)不完全等同. Danielsson 和 Hammerin[12]已经展示过,在相同阶内插函数的情况下,可分算法产生的结果中引入的噪声,比传统算法要大. 为了减小误差,可以用更加复杂的内插方法,比如高阶多项式或 B 样条法,但这增加了计算的复杂度. 他们并没有详细地分析这些额外的误差从何而来. 本节将给出可能的误差来源,并阐述如何避免它们.

通过观察可分解过程的式(4)和式(1a)可知,问题在于两式中的 p 可能是不相等的. 在连续域里, q 的真实值应是下式的计算结果(由式(11)得),

$$q' = j\sec\theta + p'\tan\theta \tag{16}$$

其中 p 是式(12)的输出(y 值). 然而,由于第一次分解中并不知道 p 的真实值,用的是四舍五入后的值 p,所以式(16)变成,

$$q' = j\sec\theta + (p+e)\tan\theta$$
$$= j\sec\theta + p\tan\theta + e\tan\theta = q_1 + e\tan\theta \quad (17)$$

其中 q_1 是第一次分解计算中实际得到的值. 若 $|\theta| \le 45°$,则,

$$|q' - q_1| \le |e| \le 0.5 \quad (18)$$

图 3 给出了 q 可能的误差区域. 图 4 给出了两个例子,表示由于舍入误差 e 的影响, q 两个可能的计算值(q_1 或 q_2). 这样,如果在第一次分解计算中,只是用 T_3,就有可能在中间图像中存储了一个错误的像素值 $F(p', q_1)$. 这个误差将不可避免地扩散到下一步.

这个问题同样可以用额外的存储空间来解决. 假设第一次分解,在行方向上,存储离点(p', q_1)最近的三个像素值,比如图 4(b)中的 $F(p, q-1)$, $F(p, q)$和 $F(p, q+1)$. 当得到 q_1 的真实值,即得到 p 的值时,可在第二次分解中实现内插. 具体过程如下:

(1) 分解 1

由式(13)得,

$$q_1 = j\sec\theta + p\tan\theta = q + b' \quad (19)$$

其中 $q = \mathrm{Round}(q_1)$,即为 q_1 四舍五入后的值. 接着在三个中间图像 I_2, I_3 和 I_4 中存储三个相关的像素值 $F(p, q-1)$, $F(p, q)$和 $F(p, q+1)$. 另外还需在一个整数图像 I_1 中存储三个点的起始列 $q-1$. 所以,在中间过程,每个点(p, j)需存储四个值.

(2) 分解 2

当由式(12)得到 p 后,就可由式(16)求出 q 的真实值. 设

$$p = \lfloor p' \rfloor = \mathrm{Floor}(p') \quad (20a)$$
$$a = p' - p \quad (20b)$$

且

$$q = \lfloor q' \rfloor = \mathrm{Floor}(q') \quad (21a)$$
$$b = q' - q \quad (21b)$$

根据下面的测试结果

$$q = I_1(p, j) \ ? \text{ 和 } q = I_1(p+1, j) \quad (22)$$

从前面存储的图像 I_2 至 I_4 中,选出四个合适的点用于内插. 例如,若 $q = I_1(p, j)$且 $q = I_1(p+1, j)$,则在式(13)中,分别用 $I_2(p, j)$和 $I_3(p, j)$取代 $F(p, q)$和 $F(p, q+1)$,用 $I_2(p+1, j)$和 $I_3(p+1, j)$取代 $F(p+1, q)$和 $F(p+1, q+1)$.

若 $q = I_1(p, j)$但 $q \neq I_1(p+1, j)$,则应相应地选择 $I_2(p, j)$和 $I_3(p, j)$与 $I_3(p+1, j)$和 $I_4(p+1, j)$.

(3) 执行

A. 分解 1

定义如下的模板组,

$$\begin{cases} T_{p1} \text{ at } i, j = \{ <0, q-j, q-1> \} \\ T_{p2} \text{ at } i, j = \{ <0, q-1-j, 1> \} \\ T_{p3} \text{ at } i, j = \{ <0, q-j, 1> \} \\ T_{p4} \text{ at } i, j = \{ <0, q+1-j, 1> \} \end{cases} \quad (23)$$

和集模板

$$T\text{-set}[k:0 \text{ for } 3] = \{ T_{p2}, T_{p3}, T_{p4} \} \quad (24)$$

其中 $q = \mathrm{Round}(q_1)$.

则可由下式计算四个中间图像

$$I_1 = \mathrm{Convolve}(T_{p1}, I_{u1}) \quad (25)$$
$$\{ I_2, I_3, I_4 \} = \mathrm{Convolve}(T\text{-set}, I_g) \quad (26)$$

其中, I_{u1}是单位图像(所有的像素值为 1).

B. 分解 2

内插值来自图像集: I_2, I_3 和 I_4.

设

$$\begin{cases} c1 = q - I_1(p, j) \\ c2 = q - I_1(p+1, j) \end{cases} \quad (27)$$

其中, $p = \lfloor p' \rfloor$且 $q = \lfloor q' \rfloor$. (注意, $c1$ 和 $c2$ 只能为 1 或 0)得,

I_h at i, j

$$= (1-c1)(1-c2)\{(1-a)[(1-b) I_2(p, q) + b I_3(p, q)] + a[(1-b) I_2(p+1, q) + b I_3(p+1, q)]\}$$
$$+ (1-c1)\, c2\{(1-a)[(1-b) I_2(p, q) + b I_3(p, q)] + a[(1-b) I_3(p+1, q) + b I_4(p+1, q)]\}$$
$$+ c1(1-c2)\{(1-a)[(1-b) I_3(p, q) + b I_4(p, q)] + a[(1-b) I_2(p+1, q) + b I_3(p+1, q)]\}$$
$$+ c1\, c2\{(1-a)[(1-b) I_3(p, q) + b I_4(p, q)] + a[(1-b) I_3(p+1, q) + b I_4(p+1, q)]\}$$

或, I_h at i, j

$$= I_2(p, q)[(1-c1)(1-c2)(1-a)(1-b) + (1-c1)\, c2(1-a)(1-b)] + I_2(p+1, q)[(1-c1)(1-c2)\, a(1-b) + c1(1-c2)(1-b)]$$
$$+ I_3(p, q)[(1-c1)(1-c2)(1-a)\, b + (1-c1)\, c2(1-a)\, b + c1(1-c2)(1-a)(1-b) + c1\, c2(1-a)(1-b)]$$
$$+ I_3(p+1, q)[(1-c1)(1-c2)\, ab + c1(1-c2)(1-a)\, b + (1-c1)\, c2\, a(1-b) + c1\, c2\, a(1-b)]$$
$$+ I_4(p, q)[c1(1-c2)(1-a)\, b + c1\, c2(1-a)\, b] + I_4(p+1, q)[(1-c1)\, c2\, ab + c1\, c2\, ab] \quad (28)$$

经过一些代数运算和定义以下模板,

$$T_{a1} \text{ at } i, j = \{ <p-i, q-j, d1>, <p+1-i, q-j, d2> \}$$
$$T_{a2} \text{ at } i, j = \{ <p-i, q-j, d3>, <p+1-i, q-j, d4> \}$$
$$T_{a3} \text{ at } i, j = \{ <p-i, q-j, d5>, <p+1-i, q-j, d6> \} \quad (29)$$

其中

$$d1 = (1-c1)(1-a)(1-b)$$
$$d2 = (1-c2)\, a(1-b)$$
$$d3 = (1-c1)(1-a)\, b + c1(1-a)(1-b) = (1-a)[(1-c1)\, b + c1(1-b)]$$
$$d4 = (1-c2)\, ab + c2\, a(1-b) = a[(1-c2)\, b + c2(1-b)]$$
$$d5 = c1(1-a)\, b$$
$$d6 = c2\, ab$$

结果可表示为,

$$I_h = \mathrm{Convolve}(T_{a1}, I_2) + \mathrm{Convolve}(T_{a2}, I_3) + \mathrm{Convolve}(T_{a3}, I_4) \quad (30)$$

6 结论

本文探讨了有效地并行执行图像旋转的问题,给出了基于二维可分旋转技术的解决方案,并进行了全面的讨论. 利用文中开发的适合于可分技术的高级抽象表达式,并行图像旋转可在 CIPA 框架下简洁地表达出来,并能在分布式存储并行系统中,高效率地进行计算. 本文还指出了分解运算可能引入

的额外误差,并提出了消除这些误差的方法.

致谢　作者在此感谢英国贝尔法斯特女皇大学 D. Crookes 教授给本文提出的有益的建议.

参考文献:

[1] H C Webber. Image Processing and Transputers [M]. IOS Press,1992.

[2] A Y Zomaya (ed). Parallel Computing: Paradigms and Applications [M]. International Thomson Computer Press,1996.

[3] G X Ritter, et al. Image algebra: an overview [J]. CVGIP, V. 49, 1990:297 - 331.

[4] Brown TJ, Crookes D. A high level language for image processing [J]. Image and Vision Computing,1994,12(2):67 - 79.

[5] Crookes D, Morrow P J, McParland P J. IAL: a parallel image processing programming language [J]. IEE Proceedings, Part I,1990,137(3): 176 - 82.

[6] Crookes D, Spence I T A, Brown T J. Efficient parallel image transforms: a very high level approach [A]. in Transputer Applications and Systems, Proc. 1995 World Transputer Congress [C]. IOS Press, Sep 1995,135 - 43.

[7] D Crookes, T J Brown, Y Dong, G McAleese, P J Morrow, D K Roantree, I T A Spence. A self-optimising coprocessor model for portable parallel image processing [A]. Lecture Notes in Computer Science [C], Springer Verlag,1996,1124,213 - 6.

[8] P Morrow, D Crookes, J Brown, Y Dong, G McAleese, D Roantree, I Spence. Achieving scalability, portability and efficiency in a high-level programming model for parallel architectures [A]. Proc. UK PAR'96 [C], Springer-Verlag, ISBN 0302-9743,1996,29 - 39.

[9] E Catmull, A R Smith. 3-D transformations of images in scanline order [J]. Computer Graphics,1980,14(3):279 - 85.

[10] W K Pratt. Digital Image Processing [M]. 2nd Ed., John Wiley & Sons, NY,1991.

[11] G Wolberg. Digital Image Warping [M]. IEEE CS Press, CA,1990.

[12] P E Danielsson, M Hammerin. High accuracy rotation of images [J]. GMIP,1992, V. 54(4):340 - 4.

作者简介:

董育宁　男. 1955 年出生于南京. 副教授. 1988 年于东南大学获工学博士学位,1998 年于英国女皇大学获理学 M. Phil 学位. 1992 年作为访问学者到英国伦敦帝国理工学院工作,1993～1995 年在美国德州大学做博士后研究,1995～1998 年在英国女皇大学和伯明翰大学做研究员. 主要研究兴趣是:医学图像处理;三维重建和图像数据库技术.