一种新的灰度图像的快速矩计算算法

郑运平[1),2)] 常宜斌[1)]

[1)](华南理工大学计算机科学与工程学院 广州 510006)

[2)](宾夕法尼亚大学医学院 费城 19104-6021 美国)

摘 要 矩是描述图像特征的一个重要算子,矩计算的效率与图像表示方法直接相关.非对称逆布局的模式表示模型(Non-symmetry and Anti-packing pattern representation Model,NAM)是一个有效的模式表示模型.位平面分解(Binary-bit Plane Decomposition,BPD)是降低灰度或彩色图像复杂度的一种良好方法.鉴于传统图像块表示(Image Block Representation,IBR)的块扫描方法的缺陷,该文提出了一种新的 NAM 块扫描方法,它是一种"无偏向"的块扫描方法.基于新的 NAM 块扫描方法,该文提出了二值图像的一种新的 NAM 表示方法,其贪心准则为:每次寻找一个在当前看来最大的矩形子模式.与 IBR 表示方法相比,在不改变时间复杂度的情况下,该表示方法能进一步减少矩形子模式的总数量.基于新的 NAM 表示和 BPD 方法,通过调整参与矩计算的位平面的参数,该文给出了矩计算的一个重要定理,提出了一种新的灰度图像的快速矩计算算法,并给出了该算法的形式化描述和复杂度分析,该算法不仅能实现矩的精确计算,也能实现矩的近似计算.以惯用的"Lena"、"F16"、"Peppers"标准图像及南加州大学标准纹理库中的纹理图像(http://sipi.usc.edu/database/database.php?volume=textures)、医学图像等为测试对象,实验结果表明:在丢失 5 个低位位平面情况下,传统矩计算的平均时间分别是 Spiliotis 算法和该文算法平均时间的 35.5182 和 53.9527 倍,也即,该文算法比当前最快的 Spiliotis 算法还要快 28.87%,因而是灰度图像的一种快速矩计算算法.

关键词 矩计算;几何矩;位平面分解;非对称逆布局的模式表示模型;灰度图像;图像块表示

中图法分类号 TP391 **DOI 号** 10.11897/SP.J.1016.2017.02619

A Novel Fast Calculation Algorithm of Geometric Moments for Gray Images

ZHENG Yun-Ping[1),2)] CHANG Yi-Bin[1)]

[1)](*School of Computer Science and Engineering, South China University of Technology, Guangzhou* 510006)

[2)](*School of Medicine, University of Pennsylvania, Philadelphia*, 19104-6021, *USA*)

Abstract Moment is an important operator to describe image features. The efficiency of the moment calculation is directly related to the method of image representation. The Non-symmetry and Anti-packing pattern representation Model (NAM) is an effective pattern representation model. Binary-bit Plane Decomposition (BPD) is a good method to reduce the complexity of gray or color images. Given the defect of the block scanning method for the conventional image block representation (IBR), we propose a new NAM block scanning method, which is a "no bias" block scanning method. Based on the new NAM block scanning method, we propose a novel NAM representation for binary images, which uses a greedy guideline: each time a current maximum rectangular subpattern is found during scanning and searching. Compared with the IBR of binary

收稿日期:2015-05-26;最终修改稿收到日期:2015-11-05.本课题得到国家自然科学基金(61300134)、高等学校博士学科点专项科研基金新教师类资助课题(20120172120036)、广东省自然科学基金(S2011040005815,S2013010012515,2015A030313206,2017A030313349)、中央高校基本科研业务费专项资金(2015ZM133)、2014 年国家留学基金(201406155015)资助.**郑运平**,男,1979 年生,博士,副教授,中国计算机学会(CCF)会员,主要研究方向为医学图像处理与模式识别.E-mail:zhengyp@scut.edu.cn.**常宜斌**,男,1989 年生,硕士研究生,主要研究方向为图像处理.

《计算机学报》

images, our proposed new NAM representation of binary images can further reduce the total number of the rectangular subpatterns while retaining the same time complexity. An important theorem of the moment calculation using NAM representation and the BPD method is put forward by adjusting the parameter of the bit planes that participate in the moment calculation. And then, we propose a novel fast calculation algorithm of geometric moments for gray images, and give a formal description and analysis of the complexity of the proposed algorithm. Our proposed new algorithm can not only calculate the accurate moments, but also can achieve the moments of approximation. By taking some idiomatic standard gray images, such as 'Lena', 'F16', and 'Peppers', some texture images from standard texture image library of the University of Southern California (http://sipi.usc.edu/database/database.php?volume=textures), and some medical images as the test objects, and by comparing our proposed calculation algorithm of geometric moments for gray images with the traditional calculation algorithm and some popular calculation algorithms, the theoretical and experimental results presented in this paper show that under the circumstance of discarding five lower bit-planes, the average execution time of the traditional moment calculation algorithm is 35.5182 and 53.9527 times of Spiliotis' algorithm and our proposed algorithm in this paper, respectively. In other words, our proposed algorithm is even 28.87% faster than the current fastest Spiliotis' algorithm, and therefore, it is a fast algorithm for moment calculation of gray images.

Keywords moment calculation; geometric moments; binary-bit plane decomposition; non-symmetry and anti-packing pattern representation model; gray images; image block representation

1 引 言

Hu首次提出不变矩以来[1],对矩计算的研究已有50余年的历史了,矩计算被广泛应用于图像处理、模式识别、计算机视觉、场景分析等领域[2-11],其中几何矩是用于推导平移、伸缩和旋转不变量的常用技术[12-16].图像表示和图像操作一起组成图像模型,矩计算的效率与图像表示直接相关.有效的图像表示方法具有两个方面的功能:(1)它能够提高图像表示的效率;(2)它能够提高图像操作的处理速度[17-19].给定一幅大小为$M\times M$的图像,函数$f(x,y)$的(p,q)阶几何矩的精确和近似定义[20]分别为

$$M_{pq}=\int_0^M\int_0^M x^p y^q f(x,y)\mathrm{d}x\mathrm{d}y \tag{1}$$

$$m_{pq}=\sum_{x=0}^{M-1}\sum_{y=0}^{M-1}x^p y^q f(x,y) \tag{2}$$

$|M_{pq}-m_{pq}|$则是精确矩与近似矩的误差,此误差的产生来自于$x^p y^q$在每一个像素的数值积分的累积近似.

1998年,Spiliotis等人[21]首次提出了二值图像的图像块表示(Image Block Representation,IBR)方法,并提出了一种基于IBR的快速矩计算方法.其基本思想是:给定一幅大小为$M\times M$的二值图像,假定该图像用IBR表示后的矩形块数为N,任意一个矩形块b的左上和右下角的坐标分别为(x_{1k},y_{1k})和(x_{2k},y_{2k}),且该块的特征函数$f(x,y)$为

$$\begin{cases}f(x,y)=1, & (x,y)\in(x_{1k},x_{2k})\times(y_{1k},y_{2k})\\ f(x,y)=0, & \text{其他}\end{cases},$$

则该二值图像的(p,q)阶几何矩定义为

$$\begin{aligned}m_{pq}&=\sum_{x=0}^{M-1}\sum_{y=0}^{M-1}x^p y^q f(x,y)\\&=\sum_{i=0}^{N-1}\sum_{x=x_{1i}}^{x_{2i}}\sum_{y=y_{1i}}^{y_{2i}}x^p y^q\\&=\sum_{i=0}^{N-1}\Big(\sum_{x=x_{1i}}^{x_{2i}}x^p\Big)\Big(\sum_{y=y_{1i}}^{y_{2i}}y^q\Big)\end{aligned} \tag{3}$$

令$\xi(x_{1i},x_{2i},p)=\sum_{x=x_{1i}}^{x_{2i}}x^p$,$\xi(y_{1i},y_{2i},q)=\sum_{y=y_{1i}}^{y_{2i}}y^q$,通过使用如下公式,则$\xi(x_{1i},x_{2i},p)$可在$O(1)$时间内计算出来.

$$\xi(x_{1i},x_{2i},p)=[(x_{2i}+1)^{p+1}-x_{1i}^{p+1}-(x_{2i}-x_{1i}+1)-\binom{p+1}{1}\xi(x_{1i},x_{2i},1)-\binom{p+1}{2}\xi(x_{1i},x_{2i},2)-\cdots-\binom{p+1}{p-1}\xi(x_{1i},x_{2i},p-1)]/(p+1) \quad (4)$$

其中 $\forall p\in Z^+$，$\binom{i}{j}=i!/(j!(i-j)!)$. 同理，$\xi(y_{1i},y_{2i},q)=\sum_{y=y_{1i}}^{y_{2i}}y^q$ 也可在 $O(1)$ 时间内计算出来. 因此，对含有 N 个同类块的 IBR 表示方法来说，m_{pq} 的计算复杂度为 $O(N)$.

2000 年，Flusser[22] 给出了上述任意一个矩形块 b 的精确计算式(5)，并且该公式的幂函数还可以使用备忘录方法进行提前计算. 理论与实践证明，这种方法求得的矩的精确度不仅比传统方法更高，而且通过预计算技术，它还能够显著提高计算速度.

$$M_{pq}^b=\int_{x_{1k}-0.5}^{x_{2k}+0.5}\int_{y_{1k}-0.5}^{y_{2k}+0.5}x^p y^q f(x,y)\mathrm{d}x\mathrm{d}y=\frac{(x_{2k}+0.5)^{p+1}-(x_{1k}-0.5)^{p+1}}{p+1}\times\frac{(y_{2k}+0.5)^{q+1}-(y_{1k}-0.5)^{q+1}}{q+1} \quad (5)$$

同年，Chung 等人[23] 对 S-树和 Gouraud 阴影法进行了研究，提出了灰度图像的 S-树编码(S-Tree Coding，STC)表示方法. 2005 年，Chung 等人[24] 提出了一种基于 STC 的灰度图像的矩计算算法，其复杂度为 $O(K)$，K 是图像用 STC 表示后的同类块总数. 该算法的误差受到质量控制参数 ε 的约束，ε 越大，同类块越少，矩计算速度也就越快，当然误差也越大，反之亦然. 然而即使当 ε 为 0，这种算法得到的矩值仍然是近似值. 由于 STC 表示方法是一种对称性的分割方法，分割方法受到较大限制，因此不是灰度图像的最优表示方法. 受 Packing 问题启发，为寻求图像模式的最优或近似最优的非对称分割方法，以非对称逆布局的模式表示模型(Non-symmetry and Anti-packing pattern representation Model，NAM)为研究对象，本课题组提出了一系列的基于 NAM 的图像表示和操作方法[18,25-27]. 文献[18]中的矩形 NAM 编码(Rectangular NAM Coding，RNAMC)表示方法即是灰度图像的一种有效表示方法. 2011 年，基于 RNAMC 表示方法，笔者提出了一种基于 RNAMC 的灰度图像的快速矩计算算法[27]. 通过对“Lena”、“F16”和“Peppers”等灰度图像进行测试，实验结果表明：在保持图像质量的前提下(如当 ε=20)，基于 RNAMC 表示的矩计算算法比流行的基于 STC 表示的矩计算算法快 26.63%. 同样，即使当 ε 为 0，该文得到的矩值也仍然是近似值.

在灰度图像的精确矩计算方面，2007 年，Hosny[28] 通过使用单项式的数学积分(与傅里叶变换类似)，移除了传统计算方法中的数值近似误差，提出了一种精确的几何矩计算算法，但是该算法要求将所有像素的横纵坐标限定到区间[−1,1]进行计算. 尽管 IBR 算法是二值图像的一种有效表示方法，但 IBR 算法的一个显著缺陷就是它不适合直接应用到灰度图像领域[29-30]，因为灰度图像相邻之间的像素一般是互不相同的. 2008 年，Papakostas 等人[29] 通过对传统的 IBR 方法进行改进并移植到灰度图像中，提出了一种基于灰度切片表示(Intensity Slice Representation，ISR)的灰度图像表示方法，同时给出了一种高效且精确的基于 ISR 的几何矩计算方法. 该方法主要是通过一遍扫描将一幅灰度图像转化为最多 2^{m-1} 幅灰度图像(m 为灰度图像的位深)，每幅灰度图像具有相同灰度值，但任何 2 幅灰度图像的灰度值均互不相同，然后再利用二值图像的 IBR 方法计算矩值. 2011 年，Spiliotis 等人[30] 通过使用灰度图像的二进制位平面分解(Binary-bit Plane Decomposition，BPD)及二值图像的 IBR 表示方法，提出了一种参数化的实时矩计算方法. 该文指出，在模式分析和识别应用领域里，误差小于 3%通常是可以接受的. 事实上，纵观以上灰度图像的精确矩计算方法[29-30]，不难发现：这些方法的核心思想均依赖于 IBR 表示方法和精确矩计算公式. 然而，正如 IBR 的提出者所言，IBR 表示方法并不是二值图像的一个最优表示方法，可能导致少一些块数的 IBR 表示方法可能需要花费更多的计算时间，最优的 IBR 表示的特征是包含尽可能少的块数[21]. 实际上，最优的 IBR 表示问题是一个 Packing 问题. 鉴于传统图像块表示(IBR)的块扫描方法的缺陷，本文提出了一种新的 NAM 块扫描方法，它是一种“无偏向”的块扫描方法. 受 Packing 问题和贪心算法的启发，为寻求二值图像模式的最优或近似最优的非对称分割方法，通过使用贪心策略和新的 NAM 块扫描方法，本文给出了二值图像的一种新

《计算机学报》

的 NAM 表示方法.与传统的 IBR 表示方法相比,在不改变时间复杂度的情况下,该表示方法能进一步减少矩形子模式的总数量.由于整个逆布局算法只需一遍扫描即可完成,因而保证了新的 NAM 表示方法的高效性.BPD 方法是降低灰度或彩色图像复杂度的一种有效方法.因此,基于新的 NAM 表示和 BPD 方法,通过调整参与矩计算的位平面的参数,本文提出了一种新的灰度图像的快速矩计算算法,该算法不仅能实现矩的精确计算,也能实现矩的近似计算.实验结果表明:在丢失 5 个低位位平面情况下,传统矩计算的平均时间分别是 Spiliotis[30] 算法和本文算法平均时间的 35.5182 和 53.9527 倍.也即,本文算法比当前最快的 Spiliotis[30] 算法还要快 28.87%,因而是灰度图像的一种快速矩计算算法.

本文的主要贡献为:

(1)提出了一种新的块扫描方法,即 NAM 块扫描方法.它不同于 IBR 块扫描方法,是一种"无偏向"的块扫描方法.

(2)基于新的 NAM 块扫描方法,提出了一种新的二值图像的 NAM 表示方法.与 IBR 表示方法相比,在不改变时间复杂度的情况下,该表示方法能进一步减少矩形子模式的总数量.

(3)基于新的 NAM 表示和 BPD 方法,通过调整参与矩计算的位平面的参数,给出了矩计算的一个重要定理,提出了一种新的灰度图像的快速矩计算算法.同时给出了该算法的形式化描述和复杂度分析,该算法不仅能实现矩的精确计算,也能实现矩的近似计算.与传统算法、Chung[24]、Zheng[27]、Honsy[28]、Papakosta[29]、Spiliotis[30] 等算法相比,理论分析和实验结果证明了本文算法的有效性.

(4)指出了 Chung[24] 和 Zheng[27] 算法当质量控制参数 $\varepsilon=0$ 时,其矩计算算法是病态的,不适合与精确矩计算的性能进行比较.

本文第 2 节重点介绍本文二值图像模式的 NAM 表示方法和灰度图像的 BPD 方法;第 3 节指明 IBR 块扫描方法的缺陷并提出一种新的 NAM 块扫描方法;第 4 节给出本文算法的理论基础、算法的形式化描述及时间复杂度分析;第 5 节通过与传统算法、Chung[24]、Zheng[27]、Honsy[28]、Papakosta[29]、Spiliotis[30] 等算法在性能上进行比较,从实验的角度验证了本文算法的有效性;第 6 节是结论.

2 图像的 NAM 表示和 BPD 方法描述

2.1 NAM 表示

Packing 问题在科学研究和生产实践中均有意义.NAM 是 Packing 问题的一个反问题,有关 NAM 的基本思想和抽象描述可参考文献[26].本小节重点介绍二值图像的 NAM 表示方法.

假定对于二值图像而言,所有白色像素值均为 1,黑色像素值均为 0.由于该表示方法需要应用到矩计算中,因此只需要对图像中的白像素,也即像素值为 1 的像素进行逆布局.二值图像的 NAM 表示方法的基本原理是:对于给定一个二值图像模式和一个预先定义的不同形状的矩形子模式,采用贪心算法从二值图像模式中提取出这些子模式,再用这些子模式的集合来表示给定的模式,其中,本文贪心算法的准则是:每次寻找一个在当前看来最大的矩形子模式.根据贪心算法的原理,新的二值图像的 NAM 表示方法获得的最终子模式的集合可能就是最优解,也有可能是最优解的一个很好的近似.

2.2 图像的 BPD 方法

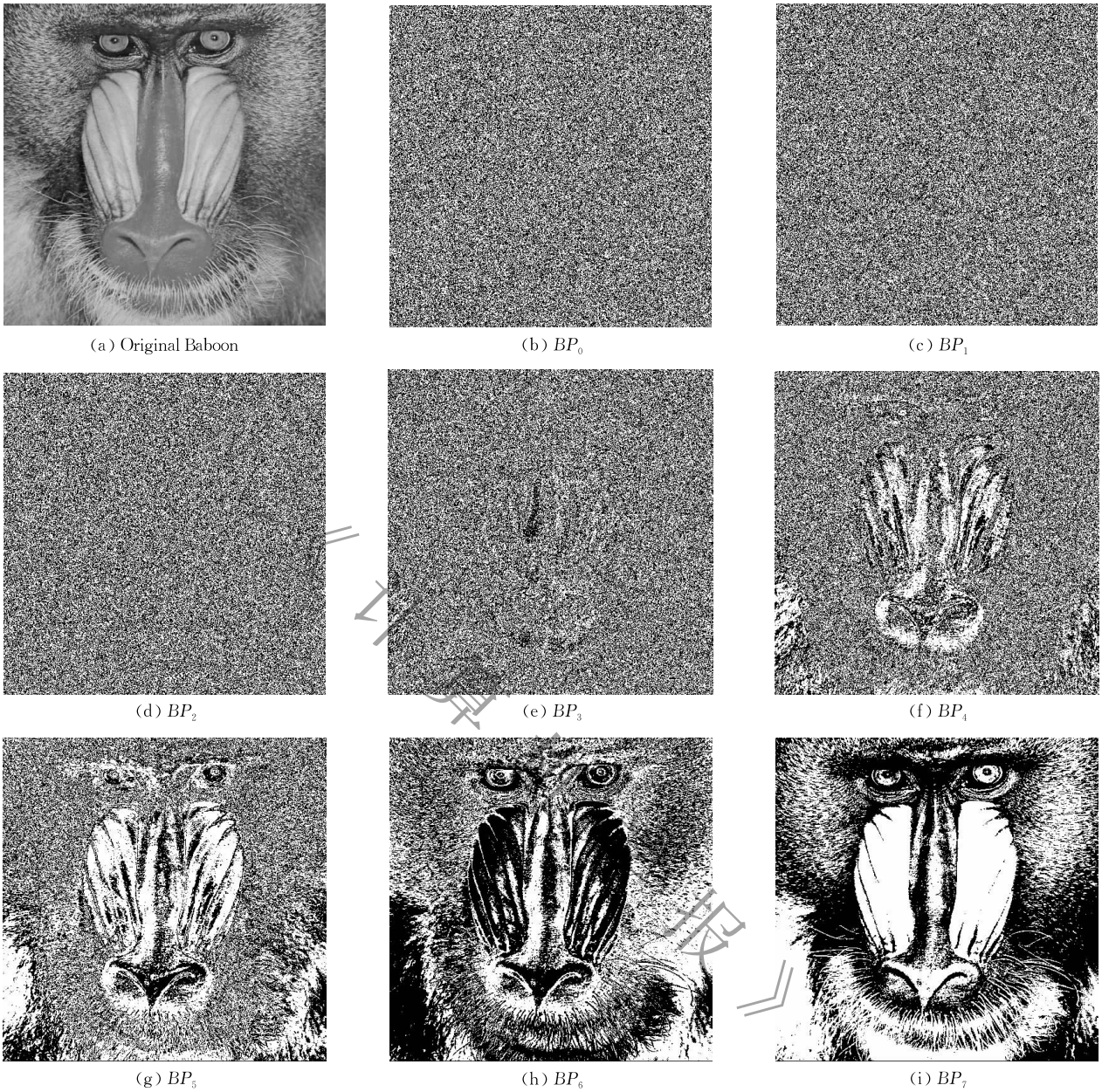
BPD 是指将一幅位深为 m bit 的灰度图像分解成 m 幅 1 bit 的二值图像.该灰度图像的灰度值可用如下多项式表示:

$$a_{m-1}2^{m-1}+a_{m-2}2^{m-2}+\cdots+a_1 2^1+a_0 2^0,$$

其中 a_i 值为 0 或 1.因此,根据这个表示,把一幅灰度图像分解成 m 幅二值图像的方法就是把上述多项式的 m 个系数分别分到 m 个 1 bit 的位平面中.

BPD 方法是一种通过单独地处理图像的位平面来减少像素间冗余的有效技术,能够有效地降低灰度或彩色图像复杂度,有关灰度图像的 BPD 方法的具体描述可参考文献[26],这里不再累述.

以位深 $m=8$、大小为 512×512 的灰度图像 Baboon 为例(如图 1(a)所示),图 1(b)~(i)给出了这种方法的示意图.图 1(a)经 BPD 方法后得到了与其相对应的 8 幅二值图像,依次用 BP_0,BP_1,…,BP_{m-1} 表示.



(a) Original Baboon　(b) BP_0　(c) BP_1

(d) BP_2　(e) BP_3　(f) BP_4

(g) BP_5　(h) BP_6　(i) BP_7

图 1　Baboon 经过 BPD 方法分解后的 8 幅位平面二值图像

3　新的 NAM 块扫描方法

本节首先简单介绍了 IBR 最核心的算法思想，然后分析了 IBR 块扫描方法的缺陷，最后提出了一种新的 NAM 块扫描方法，并将其应用到了二值图像的 NAM 表示方法中.

3.1　二值图像的 IBR 块扫描方法

假定对于二值图像而言，所有白色像素值均为 1，黑色像素值均为 0. IBR 最核心的思想是：用一系列平行于 X 轴和 Y 轴的矩形对源图像进行划分，同一矩形内所有像素值均为 1，这样得到的一个矩形称为一个同类块，每个块用 4 个整数描述：块左上角 X 坐标和 Y 坐标，块右下角 X 坐标和 Y 坐标，即 $b_i=(X_{1,bi},X_{2,bi},Y_{1,bi},Y_{2,bi})$，因此二值图像 $f(x,y)=\{b_i:i=0,1,2,3,\cdots,N-1\}$（其中 N 是同类块的块数）. IBR 提取块的过程并不复杂，其中并不涉及复杂的数值计算而仅仅是像素值的检查.

IBR 块扫描方法的具体步骤如下：

(1) 对第 Y_1 行遇到的一个新像素，令 $Y_2=Y_1$.

(2) 在第 Y_2 行：先从左往右找到本行最宽区间的横坐标 X_1，X_2（即矩形沿水平方向拓展）.

(3) 对下一行 (Y_2+1) 同样区间内 (X_1,X_2) 的横坐标内的像素值，检查是否符合同类块的条件（即

矩形沿垂直方向拓展).

(4) 如果步骤(3)的结果不符合,则新的块开始.

(5) 如果步骤(3)的结果符合同类块,则 $Y_2=Y_2+1$,执行步骤(2).

3.2 IBR 块扫描方法的缺陷

通过大量实验和观察,发现 IBR 方法有一个明显的缺陷,即在扫描同类块的时候采用了类似于行优先的方式,即先搜索一行中最宽的,然后再垂直向 Y 方向拓展,这种方法过于偏向水平方向,容易产生较多的"条带状"同类块,增加了同类块的数量,因而也增加了块表示的时间. 图 2 给出了文献[21]中的字母"d"的 IBR 表示结果.

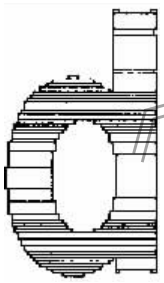


图 2 字母"d"的 IBR 表示结果

3.3 NAM 块扫描方法

本节提出了一种新的块扫描方法,即 NAM 块扫描方法,它不同于二值图像的 IBR 块扫描方法,是一种"无偏向"的块扫描方法,以下是其具体步骤:

(1) 对第 X_1 列 Y_1 行遇到的一个新像素,令 $Y_2=Y_1$, $X_2=X_1$.

(2) 如果矩形 $P_1(X_1,Y_1)$, $P_2(X_2,Y_2)$(其中 P_1 是矩形左上角坐标,P_2 是矩形右下角坐标)是同类块,则 $X_2=X_2+1$, $Y_2=Y_2+1$(即矩形沿对角线方向拓展).

(3) 如果步骤(2)的拓展结果符合同类块,重复步骤(2).

(4) 如果步骤(2)的拓展结果不符合同类块,对角方向拓展结束.

(5) 对矩形进行水平方向拓展,即 $X_2=X_2+1$.

(6) 如果步骤(5)的拓展结果符合同类块,重复步骤(5).

(7) 如果步骤(5)的拓展结果不符合同类块,水平方向拓展结束.

(8) 类似步骤(5)~(7),对此时的矩形再进行一次垂直方向拓展.

以上 NAM 块扫描方法可以保证以贪心准则找到二值图像中最大的矩形子模式集合,一定程度上降低了同类块的数目,提升了块表示的效率,是 IBR 块扫描方法的一种有效改进.

3.4 IBR 块和 NAM 块表示及扫描方法的一个实例

以 16×16 大小的二值图像 Lena 为例(如图 3 所示),图 4 给出了图像的 IBR 块和 NAM 块表示及其扫描方法的一个示意图. 不失一般性,假定图 4 中'0'和'1'元素分别表示黑色和白色像素.

图 3 16×16 大小的示例图像

在 IBR 块和 NAM 块表示方法中,我们仅需对白色像素进行表示. 图 4(a)给出了图 3 所对应的数字图像矩阵,图 4(b)给出了用 IBR 块扫描方法扫描的结果,图 4(c)给出了用 NAM 块扫描方法扫描的结果. 从图 4(b)和图 4(c)不难看出,Lena 图像用 IBR 块和 NAM 块表示所需的同类块数分别为 36 个和 32 个. 显然,在同类块数目方面,NAM 块表示比 IBR 块表示下降了 11.11%,从而更有利于节省存储空间和提高图像操作的处理速度.

用 IBR 块扫描方法和 NAM 块扫描方法表示图像时,均只需对二值图像的像素做检查,没有复杂的计算量,对规模为 M 的图像来说,两种方法的复杂度是相同的,均为 $O(M)$.

4 基于新的 NAM 表示和 BPD 的灰度图像的矩计算算法

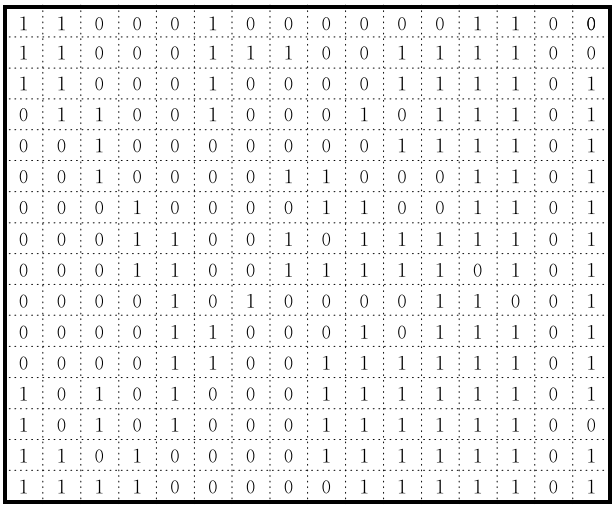
借助于文献[30]提出的 2 个重要引理,通过使用 NAM 块扫描方法和二值图像新的 NAM 表示方法,本节提出了一种基于新的 NAM 和 BPD 的灰度图像的矩计算算法,并给出了矩计算的一个重要定理.

4.1 算法的理论基础

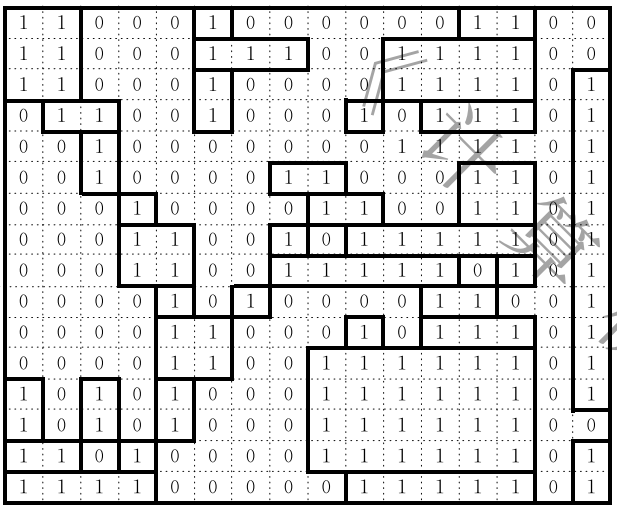
受 Packing 问题和贪心算法的启发,为寻求二值图像模式的最优或近似最优的非对称分割方法,本节给出了二值图像的 NAM 表示的一个贪心准则,即:每次寻找一个在当前看来最大的矩形子模式. 整个二值图像的 NAM 表示只需一遍扫描即可完成,因而保证了 NAM 表示方法的高效性. 另外,根据贪心算法的原理,新的 NAM 表示方法获得的最终子模式的集合可能就是最优解,也有可能是最优解的一个很好的近似. 下面给出了矩计算的几个相关定义及定理.

定义 1. 全灰度图像. 它是指图像中所有像素

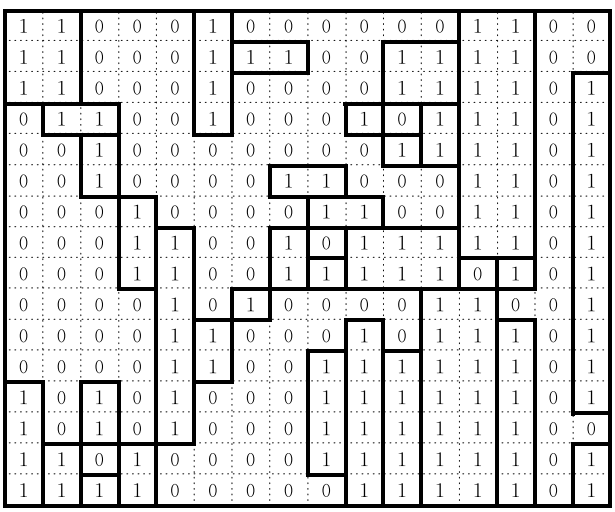
《计算机学报》



(a) 图3所对应的数字图像矩阵



(b) IBR块表示的结果



(c) NAM块表示的结果

图 4　IBR 和 NAM 块表示及扫描方法的一个实例

值均为 1 的图像.

定义 2.　棋盘图像. 它是指二值图像中任何两个相邻像素其灰度值都互不相同的图像.

定义 3.　半灰度图像. 它是指其矩值等于全灰度图像矩值一半的图像,如棋盘图像.

引理 1[30].　图像中所有像素值均为 0.5 的图像的矩值等于全灰度图像矩值的一半.

引理 2[30].　棋盘图像的矩值近似等于全灰度图像矩值的一半.

定理 1.　给定一幅位深为 m、大小为 $M\times M$ 的灰度图像,设丢弃的位平面数量为 $miss$,其中 $0\leq miss\leq m-1$. 灰度图像经 BPD 分解后参与矩计算的所有二值图像用新的 NAM 表示后的编码结果 $Q=\{R_k,k=1,2,\cdots,N\}$,R_k 为第 k 个矩形子模式的编码结果,假定该子模式的左上和右下角的坐标分别为 (x_{1k},y_{1k}) 和 (x_{2k},y_{2k}). 则基于新的 NAM 表示和 BPD 的灰度图像的矩值为

$$\begin{cases} M_{pq}=\sum_{k=1}^{N-1}\left[\frac{(x_{2k}+0.5)^{p+1}-(x_{1k}-0.5)^{p+1}}{p+1}\times\right. \\ \qquad \left.\frac{(y_{2k}+0.5)^{q+1}-(y_{1k}-0.5)^{q+1}}{q+1}\right], \quad miss=0 \\ M_{pq}\approx\sum_{k=1}^{N-1}\left[\frac{(x_{2k}+0.5)^{p+1}-(x_{1k}-0.5)^{p+1}}{p+1}\times\right. \\ \qquad \left.\frac{(y_{2k}+0.5)^{q+1}-(y_{1k}-0.5)^{q+1}}{q+1}\right]+ \\ \qquad \frac{M^{p+q+2}}{(p+1)(q+1)}\sum_{i=0}^{miss-1}2^{i-1}, \quad 0<miss\leq m-1 \end{cases}$$

证明.　假定 $g(x,y)$ 为该灰度图像中的任意一个像素,则 g 的二进制表示为

$$g=(b_{m-1}b_{m-2}\cdots b_0)_2.$$

通过使用 BPD 分解,根据文献[26]中的定理 1 即有

$$\begin{aligned} g(x,y)&=2^{m-1}b_{m-1}(x,y)+2^{m-2}b_{m-2}(x,y)+\cdots+ \\ &\quad 2^0b_0(x,y) \\ &=\sum_{i=0}^{m-1}2^ib_i(x,y). \end{aligned}$$

令 $f_i(x,y)=b_i(x,y)$,则

$$\begin{aligned} M_{pq}&=\int_0^M\int_0^M x^py^qg(x,y)\mathrm{d}x\mathrm{d}y \\ &=\int_0^M\int_0^M x^py^q\Big(\sum_{i=0}^{m-1}2^ib_i(x,y)\Big)\mathrm{d}x\mathrm{d}y \\ &=\sum_{i=0}^{m-1}2^i\Big(\int_0^M\int_0^M x^py^qb_i(x,y)\mathrm{d}x\mathrm{d}y\Big) \\ &=\sum_{i=0}^{m-1}2^i\Big(\int_0^M\int_0^M x^py^qf_i(x,y)\mathrm{d}x\mathrm{d}y\Big). \end{aligned}$$

由于位面号较低的位平面对图像的整体影响不大,可视为噪声. 根据引理 1 和引理 2,可以舍去部分低位位平面而只保留位数较高的位平面,以节省

计算机学报

计算时间,其中舍去的位平面可视为一幅白色像素和黑色像素各占一半的棋盘图像 $c_i(x,y)$,且 $0\leq i\leq miss-1$.

(1) 当 $miss=0$ 时,M_{pq} 为矩计算的精确值,此时

$$M_{pq}=\sum_{k=1}^{N-1}\left[\frac{(x_{2k}+0.5)^{p+1}-(x_{1k}-0.5)^{p+1}}{p+1}\times\frac{(y_{2k}+0.5)^{q+1}-(y_{1k}-0.5)^{q+1}}{q+1}\right].$$

(2) 当 $0<miss\leq m-1$ 时,M_{pq} 为矩计算的近似值,此时:

$$\begin{aligned}M_{pq}&\approx\sum_{i=miss}^{m-1}2^i\left(\int_0^M\int_0^M x^p y^q f_i(x,y)\mathrm{d}x\mathrm{d}y\right)+\sum_{i=0}^{miss-1}2^i\left(\int_0^M\int_0^M x^p y^q c_i(x,y)\mathrm{d}x\mathrm{d}y\right)\\&\approx\sum_{k=1}^{N-1}\left[\frac{(x_{2k}+0.5)^{p+1}-(x_{1k}-0.5)^{p+1}}{p+1}\times\frac{(y_{2k}+0.5)^{q+1}-(y_{1k}-0.5)^{q+1}}{q+1}\right]+\sum_{i=0}^{miss-1}2^{i-1}\left(\int_0^M\int_0^M x^p y^q\mathrm{d}x\mathrm{d}y\right)\\&\approx\sum_{k=1}^{N-1}\left[\frac{(x_{2k}+0.5)^{p+1}-(x_{1k}-0.5)^{p+1}}{p+1}\times\frac{(y_{2k}+0.5)^{q+1}-(y_{1k}-0.5)^{q+1}}{q+1}\right]+\frac{M^{p+q+2}}{(p+1)(q+1)}\sum_{i=0}^{miss-1}2^{i-1}.\end{aligned}$$

综上所述,可得最终的矩计算公式:

$$\begin{cases}M_{pq}=\sum_{k=1}^{N-1}\left[\frac{(x_{2k}+0.5)^{p+1}-(x_{1k}-0.5)^{p+1}}{p+1}\times\frac{(y_{2k}+0.5)^{q+1}-(y_{1k}-0.5)^{q+1}}{q+1}\right], & miss=0\\M_{pq}\approx\sum_{k=1}^{N-1}\left[\frac{(x_{2k}+0.5)^{p+1}-(x_{1k}-0.5)^{p+1}}{p+1}\times\frac{(y_{2k}+0.5)^{q+1}-(y_{1k}-0.5)^{q+1}}{q+1}\right]+\frac{M^{p+q+2}}{(p+1)(q+1)}\sum_{i=0}^{miss-1}2^{i-1}, & 0<miss\leq m-1\end{cases}$$

证毕.

4.2 基于新的 NAM 表示和 BPD 的灰度图像的矩计算算法

输入:一幅位深为 m、大小为 $M\times M$ 的灰度图像 G、丢弃的位平面数量为 $miss$,其中 $0\leq miss\leq m-1$

输出:灰度图像 (p,q) 阶几何矩的矩值 M_{pq}

1. 将用于计算矩形子模式的计数变量 N 赋值为 0.

2. 把灰度图像 G 用位平面分解方法分解为 $m-miss$ 幅二值图像 $BP_i(miss\leq i\leq m-1)$,并把这些图像连续存储;把位面号 i 赋值为 $miss$.

3. 从二值图像 BP_i 的当前入口开始,按 3.3 节提出的新的 NAM 块扫描方法,首先找到一个未访问过的矩形子模式的起始点坐标 (x,y),再根据本文提出的贪心准则来追迹相应的矩形子模式.

4. 根据矩形子模式的面积,找到一个在当前看来最大的矩形子模式,然后将该子模式在 BP_i 中标记为已访问,以方便下一个子模式的布局.

5. 令 $N=N+1$,并将此矩形子模式的左上坐标 (x_1,y_1) 和右下角的坐标 (x_2,y_2) 存储到队列 Q_i 中,即 $Q_i\{N\}\leftarrow\{(x_1,y_1),(x_2,y_2)\}$.

6. 执行步 3 到步 5,直到不能找到新的起始点为止.

7. 令 $i=i+1$. 如果 $i\leq m-1$,执行步 3.

8. 输出矩形子模式的编码结果 $Q=\{Q_{miss},Q_{miss+1},\cdots,Q_{m-1}\}$.

9. 令 $k=0$,$M_{pq}=0$.

10. k++.

11. 从编码结果 $Q=\{R_k:k=1,2,\cdots,N\}$ 中获取第 k 个矩形子模式 R_k,假定该子模式的左上和右下角的坐标分别为 (x_{1k},y_{1k}) 和 (x_{2k},y_{2k}),则该矩形子模式的 (p,q) 阶几何矩 M_{pq}^k 可通过式(6)算出,其中计算 x^p 和 y^q 可通过备忘录方法使用查找表进行预计算.

$$\begin{aligned}M_{pq}^k&=\int_{x_{1k}-0.5}^{x_{2k}+0.5}\int_{y_{1k}-0.5}^{y_{2k}+0.5}x^p y^q f(x,y)\mathrm{d}x\mathrm{d}y\\&=\frac{(x_{2k}+0.5)^{p+1}-(x_{1k}-0.5)^{p+1}}{p+1}\times\frac{(y_{2k}+0.5)^{q+1}-(y_{1k}-0.5)^{q+1}}{q+1}\end{aligned}\tag{6}$$

12. $M_{pq}=M_{pq}^k+M_{pq}$.

13. 如果 $k\leq N$,执行步 10.

14. 如果 $miss=0$,则输出灰度图像几何矩的精确值 M_{pq};否则输出近似矩值 $M_{pq}\approx M_{pq}+\frac{M^{p+q+2}}{(p+1)(q+1)}\sum_{i=0}^{miss-1}2^{i-1}$.

4.3 算法复杂度分析

对一幅位深为 m、大小为 $M\times M$ 的灰度图像的来说,设矩计算阶数为 (p,q),则传统矩计算算法的复杂度为 $O(M^2(p+1)(q+1))$,而本文提出的基于新的 NAM 表示和 BPD 的灰度图像的矩计算算法的复杂度正比于同类块的块数 N 和 $(p+1)(q+1)$. 因此,本算法的时间复杂度为 $O(N(p+1)(q+1))$. 不同图像有不同的 N 值,但是对于几乎所有图像而言,有 $N\ll M^2$,而且对比 IBR 块扫描方法,本文提出的 NAM 块扫描方法由于其无偏向性能够使 N 的数目更少,从而达到节省计算时间的目的. 另外,通过预计算还可以使计算速度更快.

计算机学报

5　实验结果

本实验中机器的主要硬件和软件配置为：内存为 Kingston DDR 2GB，中央处理器为 Celeron(R) 2.4GHz，操作系统为 MS-Windows XP. 编程环境为 Visual Studio 2008，编程语言为 C++.

通过编程，分别实现了传统算法（使用式(2)直接计算）、Chung[24]、Zheng[27]、Honsy[28]、Papakosta[29]、Spiliotis[30]等算法及本文提出的算法. 对于给定的一幅位深为 m、大小为 $M\times M$ 的灰度图像，表 1 给出了以上 7 种算法的要点描述.

表 1　7 种算法的要点描述

	是否是精确矩值	灰度图像的处理方式	图像的表示方法	几何矩的计算区间
传统算法	否（无控制参数）	直接对灰度图像进行处理，而非使用 BPD 方法分解为二值图像后再进行处理	矩阵表示方法，逐像素计算	$[0,M-1]\times[0,M-1]$
Chung[24]	否（无论是否 $\varepsilon=0$）	将灰度图像用 STC 方法分为 K 个同类块后再对每一个同类块进行计算	STC 表示方法，逐块计算	$[0,M-1]\times[0,M-1]$
Zheng[27]	否（无论是否 $\varepsilon=0$）	将灰度图像用 RNAMC 方法分为 N 个同类块后再对每一个同类块进行计算	RNAMC 表示方法，逐块计算	$[0,M-1]\times[0,M-1]$
Honsy[28]	是（无控制参数）	直接对灰度图像进行处理，而非使用 BPD 方法分解为二值图像后再进行处理	矩阵表示方法，单项式的数学积分（与傅里叶变换类似）计算	$[-1,1]\times[-1,1]$
Papakosta[29]	是（无控制参数）	将灰度图像用 ISR 方法转化为最多 2^{m-1} 幅灰度图像后再进行处理	灰度图像 ISR 表示方法及二值图像的 IBR 表示方法，逐块计算	$[0,M-1]\times[0,M-1]$
Spiliotis[30]	是（$miss=0$） 否（$miss\neq0$）	使用 BPD 方法将灰度图像分解为二值图像后再进行处理	二值图像的 IBR 表示方法，逐块计算	$[0,M-1]\times[0,M-1]$
本文算法	是（$miss=0$） 否（$miss\neq0$）	使用 BPD 方法将灰度图像分解为二值图像后再进行处理	二值图像的 NAM 表示方法，逐块计算	$[0,M-1]\times[0,M-1]$

为了对灰度图像的复杂度进行度量，文献[26]通过使用灰度图像用线性四元树表示时的总块数除以灰度图像的像素总数，给出了灰度图像复杂度 Cp 的定义. 显然有 $0<Cp\leq1$. Cp 越大表明该图像越复杂，反之，Cp 越小表明该图像越简单.

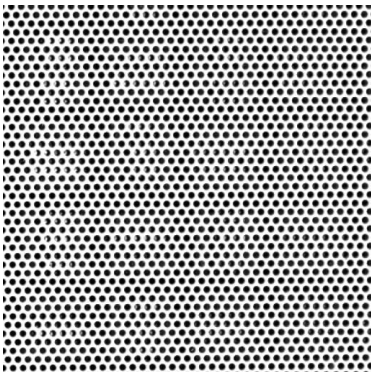
为了实验结果的可比性及比较的公平性起见，本文选取了与 Chung[24]、Zheng[27]、Papakosta[29]、Spiliotis[30]中名称相同的 3 幅图像，即“Lena”、“F16”和“Peppers”（如图 5(d)、(i)和(j)所示，但 Chung[24]没有使用“Peppers”图像）. 实验中用来测试的灰度图像的大小均为 512×512，图像的位深为 8，也即灰度级为 256 级（注：文献 Chung[24]和 Zheng[27]测试图像大小均为 512×512，而文献 Papakosta[29]和 Spiliotis[30]算法中这 3 幅图像的大小均为 200×200. 为了统一公平比较起见，本文 3 幅图像大小均为 512×512）.

此外，我们还选取了南加州大学的（http://sipi.usc.edu/database/database.php?volume=textures）标准纹理库中的纹理图像、医学图像、自然图像等 9 幅图像（大小均为 512×512），所有这些灰度图像的复杂度互不相同，具有较好的代表性，能够说明本方法的适应性.

本文将表 1 中的 7 个矩计算算法分成了 2 类，一类是精确矩计算算法，另一类是近似矩计算算法. 使用了式(1)或(5)的算法称之为精确矩计算算法，如 Honsy[28]、Papakosta[29]、Spiliotis[30]等算法及本文提出的算法，这 4 个精确算法中有的还可以通过参数控制得到矩的近似值，如 Spiliotis[30]和本文提出的算法，而另外 2 个算法由于无参数控制，只能得到矩计算的精确值；没有使用式(1)或(5)的算法称之为近似矩计算算法，如传统算法、Chung[24]和 Zheng[27]，传统算法中无参数控制，Chung[24]和 Zheng[27]都有一个误差容许度控制参数 ε，ε 越大，矩计算的误差越大，反之亦然，但即便 $\varepsilon=0$，这 2 个矩计算算法所得到的值依然是近似值，且其值与传统矩计算算法值完全一样.

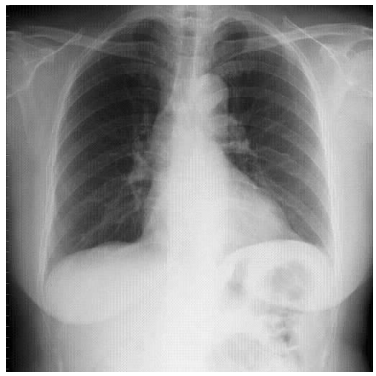
近年来，也有些学者将 $\varepsilon=0$ 时 Chung[24]的算法视作精确算法，并与他们各自所提出的精确算法在矩计算时间上进行了比较，如 Papakosta[29]和 Spiliotis[30]. 事实上，笔者认为这种比较是有失公平的，因为 Chung[24]的矩计算算法使用的是 STC 表示方法，当 ε 逐渐减少到 0 时，同类块的数量急剧上升，导致 STC 方法是相当低效的，此时利用 Chung[24]算法来计算矩是相当费时的，其计算效率甚至不如传统矩计算算法. 事实上，正如 Chung 等人[24]指出，该算法的有效性依赖于灰度图像用 STC 表示后的

计算机学报



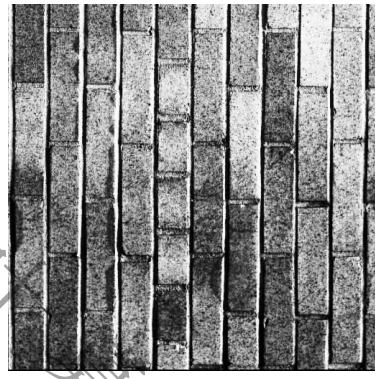
(a) Texture_1

(b) Flower

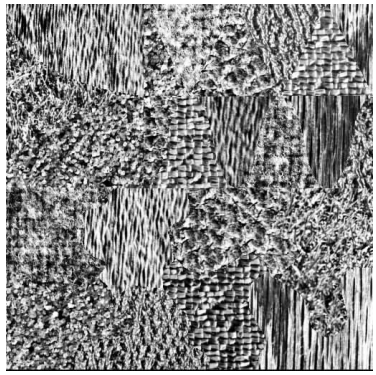


(c) Lung

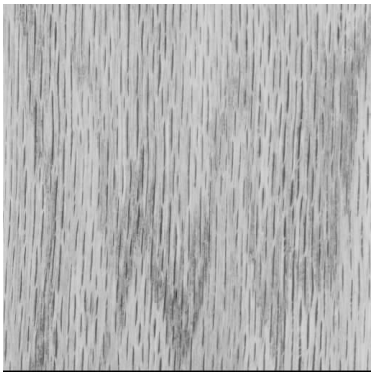
(d) F16



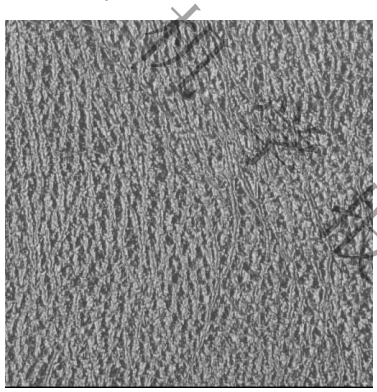
(e) Texture_2



(f) Texture_3



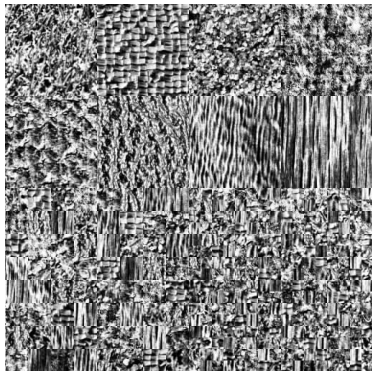
(g) Texture_4



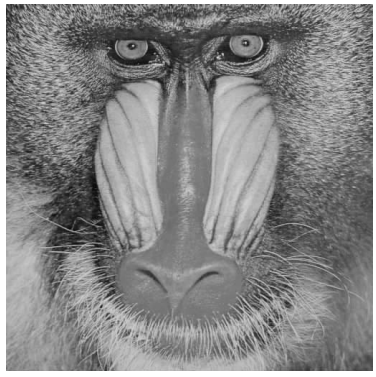
(h) Texture_5

(i) Lena

(j) Peppers



(k) Texture_6



(l) Baboon

图 5 12 幅测试图像

同类块数远小于该图像的像素总数. 因此,$\varepsilon=0$ 时 Chung[24] 的矩计算算法是病态的. 同理,Zheng[27] 的矩计算算法在 $\varepsilon=0$ 时也是病态的. 以 Lena 图像为例,表 2 给出了参数可控的 4 种矩计算算法的结果

比较. 第 2 列给出了传统算法计算的矩值，第 3 列给出了当 $\varepsilon=0$ 时 Chung[24] 或 Zheng[27] 的算法计算的矩值，第 4 列给出了当 $miss=0$ 时 Spiliotis[30] 或本文算法计算的矩值，$Diff$ 定义为第 4 列与第 2(或 3)列的差值. 从表 2 易知，当 $\varepsilon=0$ 时 Chung[24] 或 Zheng[27] 的算法计算的矩值与传统算法计算的矩值是完全一致的，均为近似值，而当 $miss=0$ 时 Spiliotis[30] 或本文算法计算的矩值与传统算法计算的矩值却不是完全一致的，$Diff$ 表明了这两者之间的差值.

表 2　4 种参数可控的矩计算算法的矩值比较

	传统矩计算	Chung[24] 算法或 Zheng[27] 算法($\varepsilon=0$)	Spiliotis[30] 算法或本文算法($miss=0$)	$Diff$
m_{00}	32 515 895	32 515 895	32 515 895	0
m_{01}	8 046 033 703	8 046 033 703	8 046 033 703	0
m_{02}	2 696 681 596 975	2 696 681 596 975	2 696 684 306 633	2 709 658
m_{03}	1 024 584 212 097 619	1 024 584 212 097 619	1 024 586 223 606 045	2 011 508 426
m_{10}	8 666 505 198	8 666 505 198	8 666 505 198	0
m_{11}	2 187 641 677 334	2 187 641 677 334	2 187 641 677 334	0
m_{12}	737 440 762 562 148	737 440 762 562 148	737 441 484 770 914	722 208 766
m_{13}	280 300 318 782 589 760	280 300 318 782 589 760	280 300 865 693 009 090	546 910 419 330
m_{20}	3 011 323 643 248	3 011 323 643 248	3 011 326 352 906	2 709 658
m_{21}	770 998 793 926 280	770 998 793 926 280	770 999 464 429 090	670 502 810
m_{22}	260 649 286 039 701 180	260 649 286 039 701 180	260 649 761 707 029 630	475 667 328 450
m_{23}	98 912 970 676 114 342 000	98 912 970 676 114 342 000	98 913 248 807 998 226 000	278 131 883 884 000
m_{30}	1 162 070 434 661 574	1 162 070 434 661 574	1 162 072 601 287 874	2 166 626 300
m_{31}	300 338 805 868 721 410	300 338 805 868 721 410	300 339 352 779 140 740	546 910 419 330
m_{32}	101 539 635 378 827 460 000	101 539 635 378 827 460 000	101 539 916 578 401 680 000	281 199 574 220 000
m_{33}	38 405 913 149 808 361 000 000	38 405 913 149 808 361 000 000	38 406 058 309 726 209 000 000	145 159 917 848 000 000

对于 $M\times M$ 大小的灰度图像来说，Hosny 通过使用类似傅里叶变换的思想，提出了一种计算矩的快速算法，但是该算法要求将所有像素的横纵坐标限定到区间$[-1,1]$进行计算，而不是通常的$[0,M-1]$. 因此，该算法计算出的矩值与本节提及的其余 6 个算法计算出的矩值无法进行横向比较. 以 Lena 图像为例，表 3 给出了 Hosny[28] 矩计算算法的矩值比较，表中第 2 列“DEM”表示“Direct Exact Method”，指原文中提及的直接精确方法[28]，第 3 列“ZOA”表示“Zeros-Order Approximation”，也即原文中的零阶近似方法[28]. 第 2 列与第 4 列的矩值是一样的，且均为精确矩，而第 3 列给出的则是近似矩，第 5 列的 $Diff$ 给出了精确矩与近似矩之间的差值.

表 3　Hosny[28] 矩计算算法的矩值比较

	DEM	ZOA	Honsy[28]	$Diff$
m_{00}	496.1532	496.1532	496.1532	0
m_{01}	21.3798	21.3798	21.3798	0
m_{02}	164.2359	164.2353	164.2359	0.0006
m_{03}	8.0125	8.0124	8.0125	0.0001
m_{10}	−15.6032	−15.6032	−15.6032	0
m_{11}	9.3676	9.3676	9.3676	0
m_{12}	−1.6962	−1.6962	−1.6962	0
m_{13}	4.7823	4.7823	4.7823	0
m_{20}	164.7991	164.7985	164.7991	0.0006
m_{21}	4.0596	4.0596	4.0596	0
m_{22}	52.4469	52.4465	52.4469	0.0004
m_{23}	−0.9475	−0.9476	−0.9475	0.0001
m_{30}	−8.2009	−8.2009	−8.2009	0
m_{31}	4.3899	4.3899	4.3899	0
m_{32}	−2.0913	−2.0913	−2.0913	0
m_{33}	1.9503	1.9503	1.9503	0

表 4 和表 5 分别给出了各种矩计算算法的同类块数和矩计算的时间比较，其中时间的单位为 ms，$Average$ 表示 12 幅图像的同类块数或矩计算时间的平均值，Cp 表示灰度图像的复杂度. 从表 4 可知，这 12 幅图像的复杂度各不相同，图 5(a)的复杂度为 0.8301，是这些图像中复杂度最低的图像，图 5(l)的复杂度为 0.9999，几乎接近 1，是这些图像中复杂度最高的图像. 对 12 幅灰度图像而言，传统算法的像素总数为 262 144，而 Papakosta[29] 算法、Spiliotis[30] 算法($miss=0$)、本文算法($miss=0$)由于分别使用了灰度切片表示和 BPD 方法，其二值图像的同类块数的平均值分别为 214 107、326 143、309 945. 而 Chung[24] 算法($\varepsilon=0$)和 Zheng[27] 算法($\varepsilon=0$)的同类块数的平均值分别为 64 837 和 64 368，均小于 Papakosta[29] 算法、Spiliotis[30] 算法($miss=0$)、本文算法($miss=0$)和传统算法的同类块数，这是因为 Chung[24] 算法($\varepsilon=0$)和 Zheng[27] 算法($\varepsilon=0$)是直接对灰度图像进行 STC 表示和 RNAMC 表示，因而具有相对少一些的同类块数，但由于这些块的复杂性，在随后的矩计算算法中将付出很大的计算代价. 从表 5 可知，当 $\varepsilon=0$ 时 Chung[24] 和 Zheng[27] 算法的矩计算的计算时间几乎是传统算法的 5 倍多，完全失去了矩计算算法的效率.

《计算机学报》

表 4 各种矩计算算法的同类块数比较

Image	Cp	传统算法	Chung[24]算法 ($\varepsilon=0$)	Zheng[27]算法 ($\varepsilon=0$)	Papakosta[29]算法	Spiliotis[30]算法 ($miss=0$)	本文算法 ($miss=0$)
Texture_1	0.8301	262144	62064	60982	172075	266358	262978
Flower	0.8860	262144	62546	57718	143051	165887	157860
Lung	0.9237	262144	63992	62770	149807	285462	273300
F16	0.9864	262144	65441	66314	204944	257466	249163
Texture_2	0.9962	262144	65522	65401	240122	405362	383718
Texture_3	0.9965	262144	65530	65317	240938	408418	387863
Texture_4	0.9967	262144	65531	65509	229665	316950	278254
Texture_5	0.9969	262144	65535	65597	249920	413502	388270
Lena	0.9970	262144	65273	65694	220545	293573	277611
Peppers	0.9988	262144	65535	65813	228552	306427	295023
Texture_6	0.9994	262144	65534	65578	242218	411621	390670
Baboon	0.9999	262144	65536	65696	247444	382692	374628
Average	0.9673	262144	64837	64368	214107	326143	309945

表 5 各种矩计算算法的时间比较

图像	传统算法	Chung[24]算法 ($\varepsilon=0$)	Zheng[27]算法 ($\varepsilon=0$)	Papakosta[29]算法	Spiliotis[30]算法 ($miss=0$)	本文算法 ($miss=0$)	Hosny[28]算法
Texture_1	493	2534	2490	60	36	33	32
Flower	493	2591	2391	56	23	19	32
Lung	493	2613	2563	57	39	35	32
F16	493	2735	2770	62	36	34	32
Texture_2	493	2768	2763	74	53	48	32
Texture_3	493	2772	2763	74	53	48	32
Texture_4	493	2779	2778	71	40	33	32
Texture_5	493	2786	2788	74	53	48	32
Lena	493	2708	2798	68	41	38	32
Peppers	493	2781	2793	71	43	40	32
Texture_6	493	2791	2792	74	53	49	32
Baboon	493	2796	2802	74	48	46	32
Average	493	2721	2708	68	43	39	32

就精确矩计算算法而言，从表 4 可知，Papakosta[29]算法的平均块数最少，Spiliotis[30]算法的平均块数最多，本文算法的块数在两者之间. 设矩的阶数为$(L-1,L-1)$，灰度切片数为 n，则 Papakosta[29]算法在最坏情况下会得到 255 个灰度切片，其计算复杂度为 $O(5L+0.5L^2+2n-2)$，而 Spiliotis[30]算法和本文算法的时间复杂度为均为 $O(kL^2)$，k 为这 2 种算法的同类块数. 从表 5 易知，在 Papakosta[29]算法、Spiliotis[30]算法($miss=0$)、本文算法($miss=0$)、Hosny[28]算法中，矩计算速度最慢的是 Papakosta[29]，其次是 Spiliotis[30]算法，最快的是 Hosny[28]算法，本文算法介于 Spiliotis[30]算法与 Hosny[28]算法之间. 但从后面的分析易知，当 $miss=5$，本文算法又是快于 Hosny[28]算法，是所有矩计算算法中最快的.

在模式分析和识别应用领域里，误差小于 3% 通常是可以接受的[30]. 表 6 和表 7 分别给出了丢弃

表 6 丢弃不同数量的位平面时 Spiliotis[30]算法的同类块总数

Image	$miss=0$	$miss=1$	$miss=2$	$miss=3$	$miss=4$	$miss=5$	$miss=6$	$miss=7$
Texture_1	266358	223186	180610	138577	98568	62816	32013	10577
Flower	165887	117028	75930	45160	24552	11833	5158	1596
Lung	285462	221232	173866	152722	142731	78501	31135	9991
F16	257466	197889	140525	90341	52549	27487	13503	3614
Texture_2	405362	348329	289831	231210	172738	117991	66466	23792
Texture_3	408418	350532	292881	234445	175696	118459	65260	22605
Texture_4	316950	257240	197590	138551	85271	47544	21963	4182
Texture_5	413502	353717	293898	234239	175036	117750	66535	29287
Lena	293573	234368	174996	117411	69568	36177	16994	5697
Peppers	306427	246432	186316	127168	74532	38298	16589	4496
Texture_6	411621	353187	295133	236433	177298	119767	66319	23209
Baboon	382692	322568	262545	202436	143256	89521	46562	18001
Average	326143	268809	213677	162391	115983	72179	37375	13087

计算机学报

表 7　丢弃不同数量的位平面时本文算法的同类块总数

Image	*miss*=0	*miss*=1	*miss*=2	*miss*=3	*miss*=4	*miss*=5	*miss*=6	*miss*=7
Texture_1	262978	220142	177794	136062	96804	61809	31633	10780
Flower	157860	111002	71510	42106	22722	10936	4746	1485
Lung	273300	210930	166318	146748	136650	74280	29668	10098
F16	249163	191657	136433	88071	51310	26773	13094	3453
Texture_2	383718	328650	272299	216088	160571	109045	60453	21002
Texture_3	387863	331862	276461	220168	163697	108990	58840	19979
Texture_4	278254	220650	162825	106615	60092	31173	14324	2347
Texture_5	388270	330619	272846	215331	158523	104141	56030	24016
Lena	277611	220433	163101	107748	62861	32318	15186	5157
Peppers	295023	236912	178823	121686	71032	36344	15675	4130
Texture_6	390670	334586	278796	222269	165396	110280	59872	20545
Baboon	374628	316481	258237	200129	142926	90614	48240	18882
Average	309945	254494	201287	151918	107715	66392	33980	11823

不同数量的位平面时 Spiliotis[30] 和本文算法参与矩计算的同类块总数 N. 从这 2 个表中可以看出，无论对于哪幅图像来说，丢失的位平面越多，参与矩计算的同类块数就越少.

表 8 给出了丢弃不同数量的位平面时 Spiliotis[30] 算法和本文算法的所有(p,q)阶矩的近似矩相对于精确矩的最大误差 $maxError$，其中 $0\leq p\leq 3, 0\leq q\leq 3$. 由于丢弃位平面后使用的替补图像均为半灰度图像，所以本文和 Spiliotis[30] 的 $maxError$ 始终是一致的. 显然，当 $miss=0$ 时，$maxError=0$. 当 $miss$ 取 0 到 5 的数目时，从表 8 可以看出，无论是Spiliotis[30] 算法还是本文算法，除了复杂度最低的图像 Texture_1 当$miss=5$ 时超过了 3%，所有 11 幅图像的$maxError$ 均是小于 3%，而当 $miss=6$ 时，Texture_1、Flower、F16 和 Lena 的 $maxError$ 都超过 3%，且当 $miss=7$ 时，除了 Texture_3 和 Texture_6 的 $maxError$ 仍然低于 3%，其余的 $maxError$ 均超过 3%.

此外，从表 8 不难发现，一般情况下，丢失的位平面越多，每幅图像的 $maxError$ 一般逐渐增大，但本表有 3 幅图像在某些丢失的数量上有例外，如 Lung，当 $miss$ 取 4、5、6 时，$maxError$ 分别为 1.8108%、1.0741%、1.0696%，也即，其 $maxError$ 不仅没有随 $miss$ 逐渐增大，反而逐渐减小；还有如 Texture_5(Texture_6)，当 $miss=3$ 时，$maxError=0.0721\%(0.0834\%)$，而当 $miss=4$，$maxError=0.0559\%(0.0509\%)$.

表 8　丢弃不同数量的位平面时 Spiliotis[30] 和本文算法的 *maxError*

Image	*miss*=0	*miss*=1/%	*miss*=2/%	*miss*=3/%	*miss*=4/%	*miss*=5/%	*miss*=6/%	*miss*=7/%
Texture_1	0	0.1135	0.3474	0.8303	1.7975	3.6867	7.1347	13.0311
Flower	0	0.0099	0.0157	0.1285	0.2677	0.4647	5.7028	9.9754
Lung	0	0.0474	0.1582	0.3656	1.8108	1.0741	1.0696	7.6608
F16	0	0.0019	0.0132	0.0768	0.5160	2.0528	4.1708	8.9258
Texture_2	0	0.0609	0.1594	0.3483	0.3972	1.4323	2.7497	6.4424
Texture_3	0	0.0082	0.0442	0.0831	0.1076	0.3750	0.7268	2.2726
Texture_4	0	0.0079	0.0234	0.0509	0.0714	0.9475	2.2905	3.7156
Texture_5	0	0.0130	0.0352	0.0721	0.0559	0.1799	1.9136	3.3684
Lena	0	0.0014	0.0023	0.0056	0.0726	0.8800	3.0323	7.3765
Peppers	0	0.0029	0.0043	0.0393	0.1843	0.8281	2.1257	15.4667
Texture_6	0	0.0203	0.0284	0.0834	0.0509	0.1108	0.2569	0.8969
Baboon	0	0.0027	0.0098	0.0196	0.0239	0.3934	2.0699	5.6086
Average	0	0.0242	0.0701	0.1753	0.4463	1.0354	2.7703	7.0617

以丢失 5 个位平面为例($miss=5$)，即省略 BP_0 到 BP_4 而使用半灰度图像进行替代计算，表 9 给出了 Spiliotis[30] 和本文算法近似计算时的性能比较. 其中 P 表示灰度图像总的像素数；N_1 和 N_2 分别表示 Spiliotis[30] 算法和本文算法的块数；T_1、T_2 和 T_3 分别表示传统算法、Spiliotis[30] 算法和本文算法的矩计算时间，单位为 ms；$R_1=T_1/T_2$，$R_2=T_1/T_3$. 以 Lena 图像为例，表 10 给出了$miss=5$ 时所有(p,q)阶矩的近似矩相对于精确矩的误差百分比 ε_{pq}，其中 $0\leq p\leq 3, 0\leq q\leq 3$，此时 $maxError=0.8800\%$，显然不超过 3%. 从 P、N_1 和 N_2 可知，在同类块数量方面，灰度图像的总像素数分别是 Spiliotis[30] 算法和本文

算法平均同类块数的 3.6319 和 3.9784 倍，而且本文算法比 Spiliotis[30] 算法的同类块数还要少 8.71%. 就矩计算的时间而言，从 T_1、T_2、T_3 可知，传统矩计算算法、Spiliotis[30] 算法和本文算法矩计算的平均时间分别为 493 ms、23.31 ms、16.58 ms，而且从 R_1 和 R_2 易知，传统矩计算的平均时间分别是 Spiliotis[30] 算法和本文算法平均时间的 35.5182 和 53.9527 倍，更进一步，本文算法比 Spiliotis[30] 算法还要快 28.87%.

表 9 Spiliotis[30] 和本文算法近似计算矩时的性能比较（*miss*=5）

Image	P	N_1	N_2	T_1	T_2	T_3	R_1	R_2
Texture_1	262144	62816	61809	493	17.62	12.33	27.9796	39.9838
Flower	262144	11833	10936	493	3.73	2.14	132.1716	230.3738
Lung	262144	78501	74280	493	23.64	16.36	20.8545	30.1345
F16	262144	27487	26773	493	8.59	6.17	57.3923	79.9028
Texture_2	262144	117991	109045	493	37.56	27.55	13.1257	17.8947
Texture_3	262144	118459	108990	493	41.09	30.97	11.9981	15.9186
Texture_4	262144	47544	31173	493	12.96	7.89	38.0401	62.4842
Texture_5	262144	117750	104141	493	35.85	25.72	13.7517	19.1680
Lena	262144	36177	32318	493	10.76	8.02	45.8178	61.4713
Peppers	262144	38298	36344	493	12.94	9.53	38.0989	51.7314
Texture_6	262144	119767	110280	493	43.41	29.44	11.3568	16.7459
Baboon	262144	89521	84598	493	31.54	22.80	15.6309	21.6228
Average	262144	72179	65891	493	23.31	16.58	35.5182	53.9527

表 10 Lena 图像的 Spiliotis[30] 和本文算法矩计算误差的百分比（*miss*=5）

ε_{pq}	$p=0$/%	$p=1$/%	$p=2$/%	$p=3$/%
$q=0$	0.0265	0.0208	0.0354	0.0375
$q=1$	0.4703	0.3174	0.2522	0.2033
$q=2$	0.7164	0.4711	0.3683	0.2991
$q=3$	0.8800	0.5769	0.4497	0.3691

综上所述，与传统算法、Chung[24]、Zheng[27]、Honsy[28]、Papakosta[29]、Spiliotis[30] 等算法相比，实验结果证明了本文算法的优越性及有效性.

6 结 论

矩是描述图像特征的一个重要算子，矩计算的效率与图像表示方法直接相关. 鉴于传统图像块表示(IBR)的块扫描方法的缺陷，本文提出了一种新的 NAM 块扫描方法，它是一种“无偏向”的块扫描方法. 受 Packing 问题和贪心算法的启发，为寻求二值图像模式的最优或近似最优的非对称分割方法，通过使用贪心策略和新的 NAM 块扫描方法，本文给出了二值图像的一种新的 NAM 表示方法. 与传统的 IBR 表示方法相比，在不改变时间复杂度的情况下，该表示方法能进一步减少矩形子模式的总数量. 由于整个逆布局算法只需一遍扫描即可完成，因而保证了新的 NAM 表示方法的高效性. 基于新的 NAM 表示和 BPD 方法，通过调整参与矩计算的位平面的参数，给出了矩计算的一个重要定理，提出了一种新的灰度图像的快速矩计算算法，并给出了该算法的形式化描述和复杂度分析，该算法不仅能实现矩的精确计算，也能实现矩的近似计算. 以图像处理领域里惯用的“Lena”、“F16”、“Peppers”标准图像及南加州大学标准纹理库中的纹理图像(http://sipi.usc.edu/database/database.php?volume=textures)、医学图像等为测试对象，实验结果表明：与传统算法、Chung[24]、Zheng[27]、Honsy[28]、Papakosta[29]、Spiliotis[30] 等算法相比，本文提出的基于新的 NAM 表示和 BPD 方法的矩计算算法是一种更为有效的矩计算方法. 更进一步，在丢失 5 个低位位平面情况下，传统矩计算的平均时间分别是 Spiliotis[30] 算法和本文算法平均时间的 35.5182 和 53.9527 倍. 也即，本文算法比当前最快的 Spiliotis[30] 算法还要快 28.87%，因而是灰度图像的一种快速矩计算算法. 这种方法在计算机视觉、场景分析、图像处理、模式识别等领域里具有良好的理论参考意义和实际应用价值.

致 谢 审稿人细致地审阅了文章，并在文章改进方面提出了宝贵的意见，在此表示感谢！

参 考 文 献

[1] Hu M K. Visual pattern recognition by moment invariants. IRE Transactions on Information Theory, 1962, 8(2): 179-187

[2] Flusser J, Suk T, Boldys J, Zitova B. Projection operators and moment invariants to image blurring. IEEE Transactions on Pattern Analysis and Machine Intelligence, 2015, 37(4): 786-802

[3] Karakasis E G, Papakostas G A, Koulouriotis D E, Tourassis V D. A unified methodology for computing accurate quaternion color moments and moment invariants. IEEE Transactions on Image Processing, 2014, 23(2): 596-611

[4] Liu Jin, Zhang Tian-Xu. The generalization of moment invariants. Chinese Journal of Computers, 2004, 27(5): 668-674(in Chinese)
(刘进，张天序. 图像不变矩的推广. 计算机学报，2004, 27(5): 668-674)

[5] Hu Yi-Ning, Zhou Jian, Luo Li-Min. A new PET reconstruction method based on Fourier-wavelet moment. Chinese Journal of Computers, 2007, 30(12): 2164-2172(in Chinese)
(胡轶宁，周健，罗立民. 一种基于 Fourier-小波矩的 PET 图像重建方法. 计算机学报，2007, 30(12): 2164-2172)

[6] Ryu S J, Kirchner M, Lee M J, Lee H K. Rotation invariant localization of duplicated image regions based on Zernike moments. IEEE Transactions on Information Forensics and Security, 2013, 8(8): 1355-1370

[7] Jassim W A, Paramesran R, Zilany M S A. Enhancing noisy speech signals using orthogonal moments. IET Signal Processing, 2014, 8(8): 891-905

[8] Zhang You-Gen, Wu Ling-Da, Song Han-Chen. Directional Zernike moments for rotation-free recognition of online sketched symbols. Electronics Letters, 2013, 49(16): 989-991

[9] Han J H, Chung W K. Active use of restoring moments for motion control of an underwater vehicle-manipulator system. IEEE Journal of Oceanic Engineering, 2014, 39(1): 100-109

[10] Mu Xing, Zhou Hou-Xing, Chen Kang, Hong Wei. Higher order method of moments with a parallel out-of-core LU solver on GPU/CPU platform. IEEE Transactions on Antennas and Propagation, 2014, 62(11): 5634-5646

[11] Zhu Hong-Qing, Yang Yan, Zhu Xiao-Li, et al. General form for obtaining unit disc-based generalized orthogonal moments. IEEE Transactions on Image Processing, 2014, 23(12): 5455-5469

[12] Flusser J, Suk T, Boldys J, Zitova B. Projection operators and moment invariants to image blurring. IEEE Transactions on Pattern Analysis and Machine Intelligence, 2015, 37(4): 786-802

[13] Singer M H. A general approach to moment calculation for polygons and line segments. Pattern Recognition, 1993, 26(7): 1019-1028

[14] Bors A G. Watermarking mesh-based representations of 3-D objects using local moments. IEEE Transactions on Image Processing, 2006, 15(3): 687-701

[15] Hannachi A, Mitchell D, Gray L, Charlton-Perez A. On the use of geometric moments to examine the continuum of sudden stratospheric warmings. Journal of the Atmospheric Sciences, 2011, 68(3): 657-674

[16] Sit A, Kihara D. Comparison of image patches using local moment invariants. IEEE Transactions on Image Processing, 2014, 23(5): 2369-2379

[17] Perret B, Lefèvre S, Collet C, Slezak É. Hyperconnections and hierarchical representations for grayscale and multiband image processing. IEEE Transactions on Image Processing, 2012, 21(1): 14-27

[18] Zheng Yun-Ping, Chen Chuan-Bo. Study on a new algorithm for gray image representation. Chinese Journal of Computers, 2010, 33(12): 2397-2406(in Chinese)
(郑运平，陈传波. 一种新的灰度图像表示算法研究. 计算机学报，2010, 33(12): 2397-2406)

[19] Wei Hui, Wang Xiao-Mei, Lai Loi Lei. Compact image representation model based on both nCRF and reverse control mechanisms. IEEE Transactions on Neural Networks and Learning Systems, 2012, 23(1): 150-162

[20] Kotoulas L, Andreadis I. Accurate calculation of image moments. IEEE Transactions on Image Processing, 2007, 16(8): 2028-2037

[21] Spiliotis I M, Mert Z B G. Real time computation of two-dimensional moments on binary images using image block representation. IEEE Transactions on Image Processing, 1998, 7(11): 1609-1615

[22] Flusser J. Refined moment calculation using image block representation. IEEE Transactions on Image Processing, 2000, 9(11): 1977-1978

[23] Chung K L, Wu J G. Improved image compression using S-tree and shading approach. IEEE Transactions on Communications, 2000, 48(5): 748-751

[24] Chung K L, Chen P C. An efficient algorithm for computing moments on a block representation of a grey-scale image. Pattern Recognition, 2005, 38(12): 2578-2586

[25] Zheng Yun-Ping, Yu Zhi-Wen, You Jane, Sarem Mudar. A novel gray image representation using overlapping rectangular NAM and extended shading approach. Journal of Visual Communication and Image Representation, 2012, 23(7): 972-983

[26] Zheng Yun-Ping, Chen Chuan-Bo. A color image representation method based on non-symmetry and anti-packing model. Journal of Software, 2007, 18(11): 2932-2941(in Chinese)
(郑运平，陈传波. 一种基于非对称逆布局模型的彩色图像表示方法. 软件学报，2007, 18(11): 2932-2941)

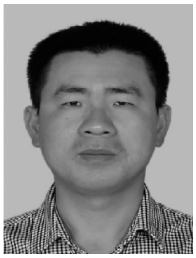
[27] Zheng Yun-Ping, Sarem Mudar. A fast algorithm for computing moments of gray images based on NAM and extended shading approach. Frontiers of Computer Science in China, 2011, 5(1): 57-65

[28] Hosny K M. Exact and fast computation of geometric moments for gray level images. Applied Mathematics and Computation, 2007, 189(2): 1214-1222

[29] Papakostas G A, Karakasis E G, Koulouriotis D E. Efficient and accurate computation of geometric moments on gray-scale images. Pattern Recognition, 2008, 41(6): 1895-1904

[30] Spiliotis I M, Boutalis Y S. Parameterized real-time moment computation on gray images using block techniques. Journal of Real-Time Image Processing, 2011, 6(2): 81-91

《计算机学报》



ZHENG Yun-Ping, born in 1979, Ph. D., associate professor. His major research interests include medical image processing and pattern recognition.

CHANG Yi-Bin, born in 1989, M. S. candidate. His major research interest is image processing.

Background

The problem researched in this paper is the moment calculation which belongs to the fields of computer vision, image processing, and pattern recognition. Moment is an important operator to describe image features. The efficiency of the moment calculation is closely related to the method of image representation. This work is supported by the National Natural Science Foundation of China under Grant No. 61300134, the Research Fund for the Doctoral Program of Higher Education of China under Grant No. 20120172120036, the Natural Science Foundation of Guangdong Province of China under Grant No. S2011040005815, No. S20130100125 15, No. 2015A030313206, and No. 2017A030313349, the Fundamental Research Funds for the Central Universities of China under Grant No. 2015ZM133, and the Chinese National Scholarship Fund under Grant No. 201406155015. The major objective of the project is to design some efficient image segmentation algorithms and some efficient moment calculation algorithms. In fact, an efficient moment calculation algorithm has a good theoretical reference significance and practical value in image processing, pattern recognition, computer vision, scene analysis and so on. Up to date, more than 50 relevant papers based on NAM for image representation and image manipulation have been published in referred conferences and journals, such as Journal of Visual Communication and Image Representation, Frontiers of Computer Science, Chinese Journal of Computers, Journal of Software, and so on. An efficient image representation method can not only improve the efficiency of image representation, but also can increase the processing speed of the image operations. An important theorem of the moment calculation using NAM representation and the BPD method is put forward by adjusting the parameter of the bit planes that participate in the moment calculation. And then, we propose a novel fast calculation algorithm of geometric moments for gray images, and give a formal description and analysis of the complexity of the algorithm. Our proposed new algorithm can not only calculate accurate moment, but also can achieve moments of approximation. By taking some idiomatic standard gray images, such as 'Lena', 'F16', and 'Peppers', some texture images from standard texture image library of the University of Southern California (http://sipi.usc.edu/database/database.php?volume=textures), and some medical images, in the field of image processing as some typical test objects, and by comparing our proposed calculation algorithm of geometric moments for gray images with the traditional calculation algorithm and Chung's algorithm (An efficient algorithm for computing moments on a block representation of a grey-scale image. Pattern Recognition, 2005, 38(12): 2578-2586), Zheng's algorithm (A fast algorithm for computing moments of gray images based on NAM and extended shading approach. Frontiers of Computer Science in China, 2011, 5(1): 57-65), Honsy's algorithm (Exact and fast computation of geometric moments for gray level images. Applied Mathematics and Computation, 2007, 189(2): 1214-1222), Papakosta's algorithm (Efficient and accurate computation of geometric moments on gray-scale images. Pattern Recognition, 2008, 41(6): 1895-1904), and Spiliotis's algorithm (Parameterized real-time moment computation on gray images using block techniques. Journal of Real-Time Image Processing, 2011, 6(2): 81-91), the theoretical and experimental results presented in this paper show that the former obviously outperform the latter, and therefore the former is a more effective fast calculation algorithm of geometric moments for gray images.