国防科技大学学报
JOURNAL OF NATIONAL UNIVERSITY OF DEFENSE TECHNOLOGY
第18卷第2期 1996年6月 Vol. 18 No. 2

一类递推关系的最佳计算格式及其在图像处理中的应用*

刘江宁 杨 嵘 王伯涛

(国防科技大学电子计算机系 长沙 410073)

摘 要 本文给出了一类递推关系在递推函数可恢复与不可恢复条件下的最佳计算格式，并讨论了它们在图像的平滑、模化及文本图像版面分析处理中的应用。

关键词 递推计算格式，图像平滑，图像模化，版面分析

分类号 TN919.8

The Best Computational Format of a Recurrence Function and Its Use in the Image Processing

Liu Jianging Yang Rong Wang Baitao

(Department of Computer Science, NUDT, Changsha, 410073)

Abstract In this paper, a best computation format of a recurrence function is given when the function is recoverable or irrecoverable. Its application to the smoothing, fattening of the images and the layout analyses of the document image is discussed too.

Key words recurrence function, smooth image, fatten image, layout analyses

设二元函数 f 满足交换律、结合律，其单位元为 e，即对于论域中任意三个元素 x，y 和 z，有

$$f(x, y)=f(y, x),\ f(x, f(y, z))=f(f(x, y), z)$$
$$f(x, e)=x$$

成立。一般地，记

$$f^{(2)}(x_1, x_2)=f(x_1, x_2)$$
$$f^{(k+1)}(x_1, \cdots, x_k, x_{k+1})=f(f^{(k)}(x_1, \cdots, x_k), x_{k+1})$$

基本问题是：对于给定的 $k(k\geqslant 3)$ 及 $\{x_j \mid 0\leqslant j<n\}$，计算由下面的 k 阶递推关系

$$y_i=f^{(k)}(x_j, \cdots, x_{j+k-1}) \qquad 0\leqslant i<n$$

* 1995年6月3日修订

所确定的集合$\{y_i|0\leqslant i<n\}$。其中，若$j<0$或$j\geqslant n$，$x_j=e$。

下面分别讨论，当函数可恢复与不可恢复的情况下，基本问题的最佳计算格式，并给出上述计算格式的图像平滑、模化及版面分析中的应用。

1 递推关系的最佳计算格式

若存在函数g，对任给的x，y，有$g(f(x,y),x)=y$成立，则称一个函数f为可恢复的；否则，称函数为不可恢复的。显然，函数$x+y$是可恢复的，而函数$\max(x,y)$，$x\vee y$，$x\wedge y$是不可恢复的。这里将x，y看作两个整数，而$\vee$、$\wedge$分析表示两个二进制整数的按位或运算及与运算。

若函数f是可恢复的，则

$$y_{i+1}=f(f^{(k-1)}(x_{j+1},\cdots,x_{j+k-1}),x_{j+k})=f(g(y_i,x_j),x_{j+k}) \tag{1}$$

从而，对每个$y_i(1\leqslant i<n)$，仅需进行一次f计算及一次g计算即可得到，而生成$\{y_i|0\leqslant i<n\}$需进行$n+k-2$次f计算和$n-1$次g计算。

若f是不可恢复的，可以利用中间结果来减少计算量。但问题在于：若格式选择不当，则可能将大量操作花费在中间量的生成上，结果反而得不偿失。为此，提出下面的计算方法。

观察$k=5$时的下列计算格式

y_i $<=(x_j(x_{j+1}(x_{j+2}(x_{j+3}\ x_{j+4}))))$

y_{i+1} $<=(\overline{x_{j+1}\ x_{j+2}\ x_{j+3}\ x_{j+4}}\ x_{j+5})$

y_{i+2} $<=(\overline{x_{j+2}\ x_{j+3}\ x_{j+4}}(\ x_{j+5}\ x_{j+6}))$

y_{i+3} $<=(\overline{x_{j+3}\ x_{j+4}}(\overline{x_{j+5}\ x_{j+6}}\ x_{j+7}))$

y_{i+4} $<=(x_{j+4}(\ \overline{x_{j+5}\ x_{j+6}\ x_{j+7}}\ x_{j+8}))$

y_{i+5} $<=\overline{x_{j+5}\ x_{j+6}\ x_{j+7}\ x_{j+8}}\ x_{j+9})$

其中，括号表示一次f计算，而上划线表示前面已经得到的中间结果。可以看出，在$k=5$时，将集合$\{y_i|0\leqslant i<n\}$划分成大小为6的组，经过12次运算可以得到每组的全部元素，且生成的中间结果可以全部利用到。当组内元素计算完毕时，中间结果刚好用完。

由此可以得到函数不可恢复时递推关系的计算格式：

(1)将n个元素$\{y[i]|0\leqslant i<n\}$划分成大小为$k+1$的组。若最后一组元素不足$k+1$个人，则需进行特殊处理。

(2)对每一组元素采用如下步骤计算：

```
begin
  p=i+k-1;q=j+k-1;y[p]=x[q];
  while (p>=i)
    begin
      p=p-1;q=q-1;
      y[p]: f(x[q], y[p+1]);
    end
```

```
    q=j+k;          z=x[q];
    for (p=i+1;p<i+k;p=p+1)
      begin
        y[p]=f(y[p], z);
        q=q+1;
        z=f(z, x[q]);
      end;
    y[i+k]=z;
  end
```

由上面的过程可以看出，每组共计算 f 函数 $3(k-1)$ 次，从而总的计算次数约为 $3(k-1)*n/(k+1)$ 次。对于每一个 y_i，平均计算次数为 $3(k-1)/(k+1)$ 次。当 k 很大时，平均计算 3 次 f 函数即可得到集合 $\{y_i|0\leqslant i<n\}$ 的一个元素值。这与每次计算 $k-1$ 次函数值来生成一个元素的朴素方法相比较，时间效率大大地得到了改善。

上述计算方法的时间复杂性与 k 的关系不大。这一事实为在图像领域中若干经典问题的处理带来了极大的好处。

2 快速图像平滑算法

图像运算是将一图像变换为另一图像的运算。设 G 为一簇数字图像，则图像运算 T 是从 G 到 G 的映射，即 $T: G\to G$。若原始图像为 g_1，变换后的图像为 g_2，上式可以写成 $g_2=T(g_1)$。图像运算可以划分为局域与全局运算两类。若对于 g_2 中的一点 (x, y)，其值 $g_2(x, y)$ 仅取决于 $g_1(x, y)$ 在邻域 $NS(x, y)$ 中各点的值，则称 T 为局域运算或局域算子。典型的局域算子有取阈值、反转、平滑、模化等。

考虑大小为 $(2N+1)\times(2N+1)$ 领域内的平滑算子 S：

$$S(x, y)=\frac{1}{M^2}\sum_{i=-N}^{N}\sum_{j=-N}^{N}P(x+i, y+j) \tag{2}$$

其中，$M=2N+1$，P 为原始图像，S 作用在 P 上的目标图像仍记为 S。当一点 (x, y) 超过图像边界时，$P(x, y)=0$。

引入水平方向和垂直方向平滑算子 Sh，Sv：

$$\mathrm{Sh}(x, y)=\frac{1}{M}\sum_{i=-N}^{N}P(x+i, y) \tag{3}$$

$$\mathrm{Sv}(x, y)=\frac{1}{M}\sum_{j=-N}^{N}P(x, y+j) \tag{4}$$

容易证明，$S=\mathrm{Sh}\circ\mathrm{Sv}=\mathrm{Sv}\circ\mathrm{Sh}$，这里“o”表示函数的合成。从而，平滑算子可以顺序分解为水平方向平滑与垂直方向平滑两个操作。两者的组合即构成 $M\times M$ 邻域内完整的平滑运算。

由加法运算的可恢复性，可对每行(列)采用公式(1)的思想进行处理，完整的平滑算法如下。其中 height、width 分别为图像的高度和宽度。

```
begin
  /* 水平平滑 */
```

```
    for (i=0;i<height;i=i+1)
      begin
        S[i][0]=0;
        for (k=-N;k<=N;k=k+1)
            S[i][0]=S[i][0]+P[i][k];
        for (j=1;j<width;j=j+1)
            S[i][j]=S[i][j-1]+P[i][j+k]-P[i][j-k];
      end
  /*垂直平滑*/
  for (j=0;j<width;j=j+1)
    begin
      /*保存第j列水平平滑结果*/
      for (i=0;i<height;i=i+1)
          q[i]=S[i][j];
      S[0][j]=0;
      for (k=-N;k<=N;k=k+1)
          S[0][j]=S[0][j]+q[k];
        for(i=1;i<width;i=i+1)
          S[i][j]=S[i-1][j]+q[i+k]-q[i-k];
  end
  /*求平均值*/
  size=(2*N+1)*(2*N+1);
  for (i=0;i<height;i=i+1)
  for (j=0;j<width;j=j+1)
        S[i][j]=S[i][j]/size;
  end
```

上述算法每生成目标图像中一个点的值需执行加、减法4次，执行除法1次。这一结果比利用(2)直接计算要优越得多。

3 快速图像模化算法

在二值图像的处理中，通常需要将图像中的一个黑点发胖为一个$(2N+1)\times(2N+1)$大小的区域。这种局域运算通常称为图像的模化(Fatten)。图像的模化结果对于两幅图像匹配度的计算、文字识别中的模板匹配及表格图像处理中的数据抽取都有着实际的意义。

设原始图像为$\{P(x, y)\}$，模化后的图像为$\{F(x, y)\}$，则模化算子可以表示成

$$F(x, y)=\bigvee_{i=-N}^{N}\bigvee_{j=-N}^{N}P(x+i, y+j) \tag{5}$$

其中，$\vee$表示max操作。在二值图像中，max可表示成位的操作。当一点(x, y)超出图像

边界时，$P(x, y)=0$。

引入水平模化和垂直模化算子 Fh，Fv：

$$\mathrm{Fh}(x, y)=\bigvee_{i=-N}^{N} P(x+i, y) \tag{6}$$

$$\mathrm{Fv}(x, y)=\bigvee_{j=-N}^{N} P(x, y+j) \tag{7}$$

易证 $F=\mathrm{Fh}\circ\mathrm{Fv}=\mathrm{Fv}\circ\mathrm{Fh}$。依照平滑算法的设计思想，可以将模化顺序划分为水平模化和垂直模化两部分来完成，所不同的是，由于 max 操作的不可恢复性，每行(列)的处理必须按不可恢复的迭代计算格式进行。

针对二值图像的特点，可以利用上述基本思想进一步提高处理速度。例如，在二值图像中，8 位组成一个字节，在进行垂直模化的过程中，只需按字节进行模化即可；在水平模化时，通过生成当前行的游程，在游程数据的基础上进行模化，然后将模化后的游程数据转换为图像位图。在表格处理软件的开发过程中，对平凡的模化算法及上述模化算法进行了比较，优化后的程序时间缩短一个数量级。当 N 越大时，上述算法的优势越明显。

有时，对图像单独进行水平模化和垂直模化也是有用的。例如在 Wahl 等设计的文本图像处理系统中[4]，文本版面的分析是在图像$\{\mathrm{Fh}(x, y)\wedge\mathrm{Fv}(x, y)\}$的基础上进行的。显然，利用本文的方法，分别计算$\{\mathrm{Fh}(x, y)\}$及$\{\mathrm{Fv}(x, y)\}$，然后对两幅图像进行与运算，即可快速生成目标图像。在文本图像的版面分析中，模化因子 N 一般都取得非常大，对于 300dpi×300dpi 的文本图像，N 的取值一般在 30～100 之间。按一般算法，在计算$\{\mathrm{Fh}(x, y)\wedge\mathrm{Fv}(x, y)\}$时，每生成一个新的像素，需计算 $4N$ 次 max 操作。由于本文所给出的方法与 N 几乎无关，对任给的 N，每生成一个新的像素只需计算 6 次 max 操作，因此同一般的模化方法相比更具有吸引力。

参考文献

1 Doster W. Designing a Document Analysis System, tutorial presented at 8th Int Conf on Pattern Recognition. Paris, FRANCE, Oct, 1986

2 Scherl W, etc. Automatic Separation of Text, Graphic and Picture Segments in Printed Material. In: Pattern Recognition in Practice, North—Holland, Amsterdam, 1980

3 Srihari, N, Zack G M. Document Image Analysis6. in: Proceedings of the 8th International Conference on Pattern Recognition, Paris 1986

4 Wahl F M, Wong K Y, Casey R G. Block Segmentation and Text Extraction in mixed Text/Image Documents. Computer Graphics and Image Processing, 1982, 20

5 Wong K Y etc. Document Analysis System. IBM J Res Dev, 1982, 26(6)

6 张系国，吴健康(译). 图像信息系统设计原理. 北京：科学出版社，1990

7 徐建华. 图像处理与分析. 北京：科学出版社，1992

(责任编辑　潘　生)