一种有效的实现分裂合并算法的数据结构*

文贡坚　王润生

(国防科技大学 ATR 国家实验室,长沙　410073)

摘　要　针对现有分裂合并分割算法数据结构存在的问题,本文设计了一种新的数据结构。这种数据结构能非常方便地实现分裂合并算法的每一步骤以及动态地分配内存,从而显著地减少了计算时间并在内存需求上有一定的改善。实验结果表明用这种数据结构实现分裂合并分割算法是有效的。

关键词　图象分割,分裂合并,数据结构,时间有效性

1　引　言

图象分割是图象分析和理解中的一项基本内容,又常常是最关键的第一步处理,其提供的区域是一个方便的、很好的图象中层描述符号,是进行图象高层理解的基础[1~2]。在众多的分割算法中,分裂合并思想以其能充分地组合利用全局与局部信息、层次式的处理结构以及较快的计算速度等优点而倍受人们的青睐。为此,人们一直在寻求一种合适的数据结构来有效地实现它,使之能获得广泛地应用。

Horowitz[3]首次将分裂合并思想用于图象分割,形成了由分裂、合并、连接编组以及小区域消除等组成的基本处理步骤,并设计了一套实现它的数据结构。由于它是基于数组形式,难于实现连接编组和小区域消除过程而使计算复杂,同时它所需内存较多。后来有较多的人对这套数据结构进行了改进,在文献[4,5]中考虑利用区域邻接图(RAG)来减少连接编组的计算量,在文献[6]中有人提议先将图象分成小块,然后在每个小块上进行分裂合并算法。

用上述固定形式的数据结构来实现分裂合并算法,无论在计算时间和内存空间上都不是有效的,人们开始研究用一种动态的结构来实现它,其中比较有影响的是 Burt 等人[7]提出了重叠四叉树结构。它是一种层次的结构,与分裂合并思想配合一致,使分裂并合中的每一步骤均在这种金字塔的数据结构中进行,正由于这一点,这种数据结构受到人们广泛地重视,并对它进行了大量的研究和改进[8,9]。但在重叠四叉树中,同样是寻找节点的相邻区域的操作比较繁琐,进行连接编组和小区域的消除是一复杂的迭代替换运算,使计算时间较长,而且执行过程中需保存整个四叉树结构,其所需内存也是相当大的。

我们认为,一种有效地实现分裂合并思想的数据结构一定要充分地发挥分裂合并思想的良好特性,即能方便地组合利用全局与局部信息、层次式的处理结构以及较快的计算速度。上面所提的数据结构均能体现分裂合并思想的前两点特性,但在计算速度方面考虑不多,使得分裂合并思想难于实用。本文在优先考虑计算速度的情况下,设计了一种新的数据结构来实现分裂合并思想。以往实现分裂合并思想的数据结构都较难实现连接编组以及小区域的消除两个过程,究其原因是在这些数据结构中,寻找节点相邻区域的算法较复杂。本文设计的数据结构是一个与图象大小一样的网格指针阵列和一些各个层次的图象块节点。在实际处理中,这种数据结构能根据待处理图象自适应地组织网格指针与相关的、不同层次的图象块节点之间的连接,从而方便、快捷地实现了分裂合并思想。基于这种数据结构,本文采用了 7 个算法模块可以完整实现分裂合并思想的全过程。这种数据结构具有以下特点:(1)它能有效地

* 国防预研基金(95J1A4.5.KG0105)和 96 卫星应用项目资助 Y96-12
收稿日期:1997-03-12

实现分裂合并思想的每一处理步骤;(2)它通过网格指针的位置主动地搜索节点的邻域,可以方便地实现现有文献中认为最复杂的连接编组以及小区域消除过程,显著地提高了计算速度;(3)它在处理过程中,根据输入图象的特性自适应地、动态地分配与释放各图象块节点,改进了对内存空间的需求;(4)它的每一处理步骤均模块化,便于扩展和修改所用的分割准则。

2　数据结构及算法实现

这种数据结构由两部分组成(图 1)。

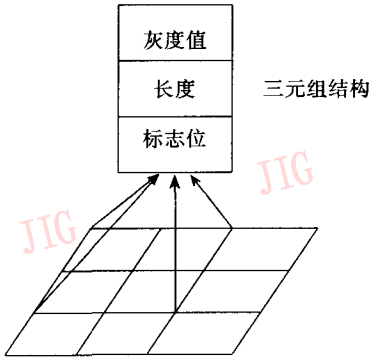

图 1

第 1 部分称为网格指针(grid pointer),它是一个和图象大小一样的指针阵列;第 2 部分是三元组的图象块节点结构,3 个元素分别为图象块节点的平均灰度值,表示不同层次图象节点块的长度以及由各算法模块决定其描述内容的标志位。网格指针和不同层次图象块节点是按下列方式联系的:除长度为 1 的图象块节点只有 1 个顶点以外,其它每个图象块节点有 4 个顶点;在网格指针阵列中,对应这 4 个顶点位置的指针分别指向该图象块节点。通过这种连接关系,可以得到一种类似于四叉树的金字塔结构。其中在网格指针中保存树的拓扑联系,长度大小不同的图象块节点表示位于不同层次的树叶节点。当然,这种数据结构并未保存整个四叉树,而只是四叉树的一部分,但可通过它来构造图象的四叉树描述。利用这种数据结构可以非常方便地实现分裂合并思想的每一步骤。

设原始图象 pI 的大小为 $N\times N$(在这里 $N=2^{l_N}$),与之大小一样的网格指针阵列为 pG,在图象的 $l_0(0\leqslant l_0\leqslant l_N)$层开始进行分裂合并,最后图象分割结果保存在图象 pO 中。基于上述提出的数据结构,我们采用 7 个算法模块来描述和实现分裂合并分割的流程。其中,算法 1 是分裂合并思想的主体流程;算法 2 是初始化树,即初始化数据结构的各单元;算法 3 是合并树,即将具有一致属性的相邻 4 个图象块节点合并为一个图象块节点;算法 4 是分裂树,即将不具有一致属性的图象块节点分裂成四个子图象块节点;算法 5 描述了整个连接编组以及小区域消除的过程;算法 6 是连接编组形成单个区域,即将相邻且具有一致属性的图象块节点连接成一个区域;算法 7 是消除小区域,即将小面积区域合并到灰度与之最接近的相邻区域中。

2.1　算法 1

```
Split-Merge//分裂合并思想的主体流程
    分配网格指针阵列 pG 并将其每一元素指向 NULL;
    InitTree; MergeTree; SplitTree; GroupingEliminate;
```

2.2 算法 2

```
InitTree        //初始化树
    num=2^l0;Step=N/num;y=0;
    do i=0 to num − 1
      do x=0, j=0 to num − 1
        分配一个图象块节点 pB,并将其长度置为 Step ,标
        志位置为 FALSE;将 pG 中 (y,x), (y, x+Step−
        1), (y+Step−1,x), (y+Step−1, x+Step−1)
        处指向 pB;x=x+Step;
      end; y=y+Step;
    end;
```

2.3　算法 3

```
MergeTree       //合并树
    l=l0−1;s=N/2^(l0−1);
    do while l≥0
      do i=0 to 2^l−1
        do j=0 to 2^l−1
          pB1=pG(is,js);pB2=pG(is,js+s/2);
          pB3=pG(is+s/2,js);
          pB4=pG(is+s/2,js+s/2);
          if 这四个节点均不为空且长度相等
            if pI 在矩形区 R(is,js,is+s−1,js+s−1)具
              有一致属性 释放 pB1,pB2,pB3,pB4,并将
              pG 中相应位置置为 NULL; 分配三元组
              pB,置长度为 s,灰度值为 pI 在 R 的平均
              值,标志位为真; 将 pG 中 R 的四个角处的
              指针指向 pB;
      end;end;end;end;
      l=l−1;s=2s;
```

```
end;
```

2.4 算法 4

```
SplitTree        //分裂树
  do i=0 to N-1
    do j=0 to N-1
      pB=pG(i,j); //找出网格指针的(i,j)图象块节点
      如果 pB 为空或其标志位为 TRUE 或其长度大于
      不执行下面操作;
      如果 pB 的长度为 1 则置其标志位为 TRUE,灰度
      值为 pI(i,j);
      if pI 在矩形块 R(i,j,i+pB 的长度-1,j+ pB 的长
        度-1)内不具有一致属性 tmpl=pB 的长度;
        释放 pB ,并将 pG 中指向 pB 的四个指针置为
        NULL;
        分别建立四个新图象块节点,置它们长度为 tmpl/
        2,标志位为 FALSE; 将 pG(i,j),(i,j+tmpl/2-
        1),(i+tmpl/2-1,j),(i+tmpl/2-1,j+tmpl/2-
        1)指向节点 1;
        ...//同样,将 pG 中对应的位置指向节点 2、3。4;
        j=j - 1;
    else
      置 pB 的标志位为 TRUE,灰度值为 pI 在 R 中的
      灰度平均值;
end;end;
```

2.5 算法 5

```
GroupingEliminate//连接编组形成区域以及小区域
的消除
  将 pO 每一元素置为 0;
  分配一块内存空间 pTemp 来保存图象块节点位置;
  blocknumber=0;//blocknumber 表示区域数目
  do i=0 to
    do j=0 to
      pB=pG(i,j);
      如果 pB 为 NULL 或 pB 的标志位为 FALSE 则不
      做;
      blocksize=GroupBlock(j,i,pTemp,blockn,gray);
        //blockn 为该连通块的节点
      //个数,blocksize 为大小,pTemp 保存节点位置,
            gray 为区域灰度值
      if (blocksize < 门限值)
        eliminatesmallregion(pTemp,blockn,gray);
                  //消除小块
      else
        将 pTemp 中所有节点值写入到输出图象 pO 中
      blocknumber=blocknumber+1;
  end;end;
```

2.6 算法 6

```
  GroupBlock (x, y, pTemp, blockn, gray)
    //连接编组形成单一区域
分配一堆栈 pS;pB=pG(y,x);PUSH(x,y,pS);
blockn=0;blocksize=0;gray=0;
while(pS 非空)
  POP(xx,yy,pS);pB=pG(yy,xx);
  if(xx 为奇数) xx=xx - pB 的长度+1;
  if(yy 为奇数) yy=yy - pB 的长度+1;
  pB 的标志位置为 FALSE;
  pTemp(blockn)置为(xx,yy);
  blocksize=blocksize+pB 的长度的平方;
  重新计算整个区域的平均灰度 gray;
  blockn=blockn+1;
  //将该块四边的图象块节点压入堆栈中,先处理上面
    一条边
  do i=xx-1 to xx+pB 的长度
    pB1=pG(yy-1,i);
    如果 pB1 等于 NULL 或 pB1 的标志位为 FALSE
    则不做;
    如果 pB1 的灰度值与 gray 的差值大于门限值则不
    做;
    PUSH(i,yy-1,pS);pB1 的标志位置为 FALSE;
  end;
  ...//同样处理其它三条边
end;
return blocksize;
```

2.7 算法 7

```
  eliminatesmallregion(pTemp,blockn,gray)
    //消除小区域
寻找小区域的外轮廓点;//外轮廓是指不属于该区域但
又紧靠该区域的点集
    biasgray=255;//biasgray 表示轮廓点灰度与 gray
的差值
for 所有外轮廓点(x,y) do
  if (pO(y,x)! =0 且 |pO(y,x) - gray|<biasgray)
    biasgray=|pO(y,x) - gray |;x0=x;y0=y;
  else if(pO(y,x)==0 且 pG(y,x)不为空且|pG(y,x)
      的灰度值- gray|<biasgray)
    biasgray =|pG(y,x)的灰度值- gray|;x0=x;y0
          =y;
end;
if(pO(y0,x0)! =0)
  do i=0 to blockn - 1
    x,y 表示 pTemp(i)中位置;pB=pG(y,x);
    将 pO 对应 pB 位置的值置为 pO(y0,x0);
```

```
    end;
  else
    do i=0 to blockn - 1
      x,y 表示 pTemp(i)中位置;pB=pG(y,x);
      将 pB 的灰度值置为 pG(y0,x0)的灰度值,并 pB 的
          标志位置为 TRUE;
    end;
```

3　实验结果

实验 1 是 Horowitz[3]和本文方法的比较,实验 2 是本文方法对一幅较大图象的分割结果,选用机器为 IBM-PC 586/100M。

实验 1:原始图象(图 2)的尺寸为,Horowitz[3]和本文方法的处理结果分别为图 3 和图 4,它们的计算时间比较列如表 1。

表 1　2 种方法计算时间比较(单位:秒)

	合并时间	分裂时间	编组与消除小区域时间	共需时间
Horowitz[3]	0.032 034	0.101 190	2.52 387	2.657 094
本文的方法	0.027 453	0.098 901	0.181 319	0.307 673
H 方法/本文方法(倍)	1.166 9	1.023 14	13.919 5	8.636 097

图 2

图 3

图 4

实验 2:原始图象(图 5)的尺寸为,由于 Horowitz[3]的方法所需内存较多,在微机上难于实现,故不作比较。本文方法的分割结果为图 6,它的计算分配时间列如表 2。

图 5

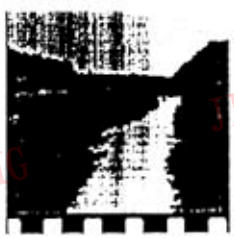

图 6

表 2　对一幅较大图象分割的时间分配(单位:秒)

合并时间	分裂时间	编组与消除小区域时间	共需时间
0.054 945	0.178 571 5	0.320 989	0.554 505 5

4　结　论

本文提出的网格指针与图象块节点相结合的数据结构能有效地实现分裂合并算法。上面的算法说明和实验结果都证实了在计算速度上它较以前方法有了显著的提高。应当指出,虽然这种数据结构在内存分配上有了较大的灵活性,但由于分裂合并分割算法比较复杂,它所需内存量还是很大的。随计算机技术的迅速发展,这不会成为实现时的主要障碍。

参 考 文 献

1　文贡坚. 图象分析新途径的研究[硕士学位论文]. 长沙:国防科技大学,1997.

2　王润生. 图象理解. 长沙:国防科技大学出版社,1995.

3　Horowitz S L, Pavlidis T. Picture segmentation by a tree traversal algorithm. J. ACM, 1976,23:368～388.

4　Chen P C, Pavlidis T. Segmentation by texture using a co-occurrence matrix and a split-and-merge algorithm. Comput. Graphics Image Process, 1979,10:172～182.

5　Wu X L. Adaptive split-and-merge segmentation based on piecewise least-square approximation. IEEE Trans. Pattern Anal. Mach. Intell., 1990,15(8):808～815.

6　Browning J D, Tanimoto S L. Segmentation of pictures into regions with a tile-by-tile method. Pattern Recognition, 1982,15:1～10.

7　Burt P J, Hong T J, Rosenfeld A. Segmentation and estimation of image region properties through cooperative hierarchical computation. IEEE Trans. Syst. Man, Cybern., 1981,11:802～809.

8 Baronti S. Variable pyramid structure for image segmentation. Comput. Vision Graph. Image Process. ,1990,49:346～356.
9 Ang C H, Samet H, Shafer C A. A new region expansion for quadtrees. IEEE Trans. Pattern Anal. Mach. Intell. , 1990, 12 (7):682～686.

文贡坚 出生于 1972 年，分别于 1994 年和 1997 年在国防科技大学电子工程学院获学士和硕士学位，现为该校 ATR 国家实验室博士研究生。现研究方向为图象符号化与特征提取、目标识别和数据融合。

王润生 1964 年毕业于军事工程学院电子工程系，现为国防科技大学电子工程院教授，博士生导师，ATR 国家重点实验室第四研究室主任，中国图象图形学会常务理事。现研究方向为图象分析与理解、模式识别和信息融合等。

An Efficient Data Structure for Realizing Split-and-Merge Method

Wen Gongjian, Wang Runsheng

(*ATR State-key Lab. National University of Defense Technology ,Changsha* 410073)

Abstract An efficient data structure for realizing spilt-and-merge image segment-ation method is devised in order to over-come the problems in the current methods. This data structure can very easily realizes each step in the idea of split-and-merge and dynamically assigns memory, it results in a remarkable decrease in computational time and an improvement in memory space. The experimental results show that this method developed is very efficient in realizing split-and-merge.

Keywords Image segmentation, Split-and-merge, Data structure, Time-efficient

单一芯片的 PC
1999 年年中可望问世

据报道，1999 年年中，将有可能把 PC 上几乎所有的工作芯片都集成到一块芯片上。所谓单一芯片 PC 就是把执行各种不同任务(如中央处理、图形处理、视频和音频处理等)的多个硅片集成到一块硅片上，最终的产品样式将是在一块大约半英寸宽的硅片上集成 10 多种处理功能。但美国国家半导体公司声称，这种单一芯片还不能把内存和电源功能集成进去。

单一芯片 PC 系统的成本肯定要比多芯片同档次系统低得多，如此一来，现在的许多基本配置 PC 的价格就可以降至 1 000 美元以下。另外，单一芯片技术令能耗大大地降低，使其更加适用于便携机和各种信息设备。

单一芯片 PC 将采用分布式处理技术，以便该芯片的各个不同部分在执行各种专门任务，如多媒体或通讯任务时能得到优化。资源的这种优化配置无疑会改善该芯片的性能。

美国国家半导体公司将在位于缅因州南波特兰市新建的波表制造厂生产这种芯片，计划每月生产 3 万片。在初期生产中，将采用 0.25 微米工艺，以后将采用 0.18 微米工艺。如果需要的话，美国国家半导体公司还将利用它的一些合作伙伴如 IBM 微电子公司的生产厂来提高产量。