快速连通域分析算法及其实现

孔 斌
(中国科学技术大学 自动化系 合肥 230027)
(中国科学院 合肥智能机械研究所 合肥 230031)

摘 要 本文提出一种快速连通域分析算法,它对像素的行程进行操作,并将标号作为行程及连通域的特征之一,特征通过数据结构的指针与行程及连通域相联系.该算法运用了两个关键技术,一是设计了一种链式机制来表示和实现标号的等价关系,二是通过指针的传递来实现标号及其它特征的向下传递和逆向传播,特征在标号过程中动态修改.这样甚至能实现仅对图像一遍扫描便能完成连通域标记和常用特征量的计算.实验表明了本文算法的有效性.

关键词 连通域分析,连通域标记,行程,链式等价机制,指针
中图法分类号 TP391

1 引 言

在许多计算机视觉应用中,常常需要进行连通域分析(CCA, Connected Component Analysis).它包括了三部分操作内容:(1)连通域标记(CCL, Connected Component Labeling),即在规定的连通性定义下,给每一个物体或目标(即所谓的连通域)分配一个唯一的非零标号;(2)计算每一个连通域的形状特征;(3)计算每一个连通域的灰度特征.其中连通域标记是后两部分计算的基础.如果设定标号为连续自然数,那么最后所得到的最大的连通域标号就是图像中所有的连通域的总数.

对连通域进行标记的一种直接而简单的方法包含搜索和传递两个步骤[1].该方法虽然简单,但极费时,因此研究者们提出了许多改进的方法[2-4].可以说,到上世纪 80 年代,连通域标记算法已经成熟;但是,近十多年来仍不断能见到与其有关的杂志及会议论文[5-14].这主要是因为大多数算法需花费较多的时间和/或存储空间,使其实际应用受到限制;另一方面是它经常被用于在多处理器系统中研究并行算法及性能[8,10-12].

近年来,随着电子技术的发展以及计算机整体性能和配置的提高,算法对存储空间的要求已较容易满足,算法运行时间也大大降低了.但是对于实际应用常常具有的实时要求,现有算法一般仍不能满足.所以,寻求快速算法仍是研究者的目标.

本文提出的算法,不仅能很快得到连通域的标号,而且也能在标号的过程中方便地计算连通域的多种常用的灰度和形状特征,标号完成时也得到各种特征统计值.因此称该算法为快速连通域分析算法(而不仅仅是快速连通域标号算法).文中将详细介绍该算法的决窍,并通过将其标号部分与三种著名的连通域标号算法的实验比较和分析来说明本文方法的高效性.

2 快速连通域分析算法

文献[1]指出,加速连通域标号执行过程的一种常用方法是对目标像素的行程(run-length)而不是对每个像素本身进行标号.快速连通域分析算法(下文中简称为 FCCA 算法)吸收了这一思想,针对目标像素的行程进行标号和等价类分析.这个算法的关键之处是把行程的标号作为行程的一个特征来处理,行程的特征通过结构指针与行程段相联系,并且令属于同一个连通域的所有行程段拥有同一个特征

收稿日期:2001-10-13;修回日期:2002-10-28

数据块.这样可以通过指针的传递来实现标号及其它特征的传递.并且,设计了一种链式机制使等价关系的表示和记录在1维数组中就能实现,也易于进行等价类划分.

下面先给出本文的FCCA算法步骤的描述,并稍加解释.

FCCA **算法:**

假设已对输入图像进行了编码(run-length encoding).

STEP 1　创建并初始化等价数组.并设LABEL=1,行程指针指向第一个行程段.

STEP 2　如果行程指针指示图像行程结束,GOTO STEP 8.

STEP 3　如果当前行程THIS在前一行中没有相邻接的行程,

3.1　为THIS创建一个新的特征块;

3.2　把LABEL赋给THIS,并计算THIS的其它特征;

3.3　LABEL=LABEL+1,行程指针加1,GOTO STEP 2.

STEP 4　如果THIS在前一行中仅有一个与它相邻接的行程PREV,则

4.1　把PREV的特征指针与THIS相联系;

4.2　修改PREV也即THIS的其它变化了的特征(比如面积);

4.3　行程指针加1,GOTO STEP 2.

[解释:这里,通过特征指针的传递,把PREV的特征包括标号传给了THIS.]

STEP 5　否则可以确定THIS在前一行中有多于一个与其相邻接的行程,假设为$P_1,P_2,\cdots,P_n$, $n\geqslant 2$,并且假设P_1是最先出现的行程.此时

5.1　把P_1的特征指针与THIS相联系;

5.2　修改P_1也即THIS的其它变化了的特征.

[解释:这种情况说明$P_1,P_2,\cdots,P_n$都属于同一个连通域,它们至少通过THIS连接成一个整体.这一步中,通过指针把P_1的特征包括标号传给了THIS.下面还必须检查$P_2,\cdots,P_n$的标号是否与P_1的一致.若不一致,应进行等价关系分析.]

STEP 6　如果$P_2,\cdots,P_n$都与P_1具有相同的标号,则行程指针加1,GOTO STEP 2.

[解释:具有相同标号的行程的特征指针已经指向同一位置.因此STEP 5.2中对P_1的特征的修改同样对$P_2,\cdots,P_n$起作用.]

STEP 7　否则$P_2,\cdots,P_n$中存在与P_1不同的标号,此时

7.1　对每一个与P_1有不同标号的$P_i(2\leqslant i\leqslant n)$,

(1) 在等价数组中记录P_i的标号与P_1的标号等价,

(2) 将P_i的特征指针改为与P_1相同的特征指针,

(3) 对其中第一次出现的那些不同标号,对P_1的特征作相应修改(即连通域合并);

7.2　行程指针加1,GOTO STEP 2.

[解释:每一次特征数据的修改,对拥有引起此修改的标号的所有行程段都起作用.由于属于同一个连通域的行程通过结构指针的指示都拥有共同的特征块,这一步隐式地实现了标号的传递和逆向传播.]

STEP 8　如有必要,根据等价数组将行程标号改为连续自然数,或进行行程解码(run-length decoding)得到标号图像. ◇

可以对本算法标号的正确性进行证明,它与Lumia算法的证明[2]类似.区别在于,本算法利用指针使所有属于同一个连通域的行程都有共同的特征块,通过指针的传递隐式地实现了标号的传递、等价和逆向传播过程.这里不再给出详细证明.

FCCA算法的另一个重要特点和优点,是它能很容易地在标号过程中同时把连通域的常用灰度、形状特征计算出来.这就是不把它仅仅称为快速连通域标号算法,而是称为快速连通域分析算法的原因.另外,如果一边扫描图像获得行程编码,一边便对所获得的行程进行标号和特征计算,那么甚至能对图像仅经一遍扫描便得到连通域分析的结果.

3　FCCA算法的实现

3.1　数据结构

图像中每段行程的结构可设计如下:

```
typedef struct {
    int begin, end; //行程起止列号
    int length;     //行程长度
    FEATURES * feature; //行程(连通域)的特征块指针
}a_run;
```

该结构有冗余是为了编程方便.在设计存储整幅图像的行程码的结构时,也应尽量使用指针.使用指针的好处除了易于传递数据外,还可以仅在必要时才分配存储空间,从而达到节约内存的目的.

连通域和行程的特征块结构可以根据具体应用所需计算的特征的性质和数目确定.若仅需做连通域标号,则特征块结构可简单地定义为:

```
typedef struct { //特征块结构之最简例:
    int Label;  //行程以及连通域的标号
} FEATURES;
```

如还要计算其它特征,则在结构中可相应地增加其它参数.须说明的是,灰度特征的计算还需以灰度图像为依据,可在行程编码时同时参考灰度图像并在每段行程的结构(a_run)中增加相应的灰度统计参数.下文中不再专门考虑灰度特征.下面是带形状参数的特征块结构的一个例子:

```
typedef struct { //特征块结构之形状参数例:
    int Label;      //行程及连通域的标号
    int npoints;    //0阶矩,面积
```

```
    int xsum, ysum; //1 阶矩
    double x0,y0;       //形心
    int xxsum, xysum, yysum; //2 阶矩
    double xx,xy,yy;    //用于计算拟合椭圆
    int left, top, right, bottom; //外接正立长方形
} FEATURES;
```

连通域和行程的特征块的另一功用是计算后直接包含了许多常用的连通域分析结果,因此如无必要就不用再进行顺序重标号以及行程解码(run-length decoding)来得到标号图像.

3.2 链式等价机制

只有当应用要求给出一幅标号图像,或需根据标号结果计算目标特征时,才必须使用等价表.为保证算法通用性,在FCCA中仍然保留了等价表,并且设计了一种链式机制实现和表示等价关系.

考虑任意一个1维数组,它有 N 个元素,并且数组元素值为不大于 N 的自然数.若数组元素的索引号与对应的元素值不等,当把该元素值看作另一个索引号时,就可得到另一个元素值.若用→表示"对应",—表示"看作",那么,索引号→元素值—索引号→元素值—索引号→元素值,就形成了一条链.这样的链将数组划分为若干个互不相交的部分,每一部分的元素值必定为该部分中的索引号之一.如果一条链中有某个索引号与对应元素值相等,那么称该索引位置为"链头",链的另一端可相应称作"链尾".如果一条链中的某对索引号与元素值曾经出现过,那么这条链上存在一个"环".可以证明,数组被链划分成的每一部分中,要么有且仅有一个链头,要么有且仅有一个环.(一种特殊情况是,某个部分仅有一个索引号,其元素值为索引号本身.根据前面的定义,称该索引位置为"链头".)

因此,若将相等价的两个标号之一设为数组元素的索引号,另一个设为相应的元素值,则数组可被等价标号链划分成各个等价类.现在问题就变为:怎样给数组元素赋值才能保证一个等价类决不会被划分成两个或多个部分?只要解决了它,链式机制便可以用来表示和记录标号等价关系.

FCCA算法中,等价表是个固定大小的1维数组,因为在具体应用中,一般对图像的复杂程度(即可能出现的最大标号)都能有一些经验的了解,从而可以确定一个适当的大小.当然,在程序运行过程中动态分配等价表的内存也是可以的,但是这样须花费一定的时间代价.

链式等价机制的具体实现策略如下:因为负数一般不作为标号使用,所以可把它们作为等价链中链头的标志,例如,取-1为标志.FCCA算法的第一步中,先把等价表数组的所有元素初始化为-1.假设在某一行中有两个不同的标号被判断为等价,那么将数组元素的索引号设为这两个标号中对应行程列号较大的那一个,元素值为另一个.例如,设等价表数组为eqTable[],某一行中依次出现标号i,j,k ($i\neq j\neq k$),先后被判明i与j等价、j与k等价,则令:eqTable[j]=i, eqTable[k]=j.同时修改该行i之前出现的j标号(以及j之前出现的k标号,详见3.4节),这样的策略保证了在等价链中一定不存在环,并且相等价的所有标号一定能通过等价链达到共同的链头.

每一段行程的最终标号为其原有标号所在等价链中链头的索引号.当需要连续自然数标号时,可将那些元素值为-1的依次改为连续自然数的相反数(负数表示链头),每一段行程的最终标号是其链头元素值的相反数(负负得正,仍是自然数).

3.3 特征计算

一般常用统计矩来表示物体的形状特征.如果令 C 表示连通域,r 表示其中的行程段,i_r 表示 r 的行号,b_r、e_r 分别表示 r 的起止列号,l_r 表示 r 的长度,那么 C 的0至2阶统计矩可按以下公式计算:

$$M_0(C)=\sum_{r\in C}l_r$$

$$M_x(C)=\sum_{r\in C}l_r\frac{b_r+e_r}{2}$$

$$M_y(C)=\sum_{r\in C}l_r i_r$$

$$M_{xx}(C)=\sum_{r\in C}l_r\left(b_r e_r+\frac{(l_r-1)(2l_r-1)}{6}\right)$$

$$M_{xy}(C)=\sum_{r\in C}l_r i_r\frac{b_r+e_r}{2}$$

$$M_{yy}(C)=\sum_{r\in C}l_r i_r^2$$

其中 x 对应列坐标,y 对应行坐标,0阶矩 $M_0(C)$ 就是 C 的面积 A. C 的形心位置 (x_0,y_0) 及其拟合椭圆的两轴长 a,b 和主轴方向 θ 计算如下:

$$x_0=\frac{M_x(C)}{A},\quad y_0=\frac{M_y(C)}{A}$$

$$a=M_{xx}(C)-x_0M_x(C)$$

$$b=M_{yy}(C)-y_0M_y(C)$$

$$c=M_{xy}(C)-x_0M_y(C)=M_{xy}(C)-y_0M_x(C)$$

$$\theta=\frac{\pi}{2}+\frac{1}{2}\arctan\left(\frac{2c}{a-b}\right).$$

如果要确定 C 的大致范围,可以用它的外接正立矩形上(U)下(D)左(L)右(R)位置来参数化:

$$U(C)=\min_{r\in C}i_r,\quad D(C)=\max_{r\in C}i_r,$$

$L(C)=\min_{r\in C} b_r$,　$R(C)=\max_{r\in C} e_r$.

上述公式在计算机上实现极为容易,这里不再赘述.

3.4　标号过程及标号的传播和等价

对行程段的标号有以下三种情况:

(1)分配新的区域标号.此时为当前行程段创建一个新的特征块,把尚未使用过的最小标号分配给当前行程段并计算其它特征,将特征指针赋给当前行程段.

(2)直接接受前一行相邻行程段的区域标号.此时将前一行相邻行程段的特征指针赋给当前行段并修改有变化的特征.标号和其它特征通过此指针传递而得到下传.

(3)如果当前行程段已被标号为 p_1,在前一行又有一个标号为 p_2 的行程段与它相邻接,说明标号 p_1 和 p_2 等价.此时应将标号为 p_1 和 p_2 的两个连通域合并.具体做法为:

首先修改当前行中在当前行程 THIS 前出现的已被标为 p_2 的行程的特征指针,为 THIS 的特征指针(即指示标号为 p_1 的特征指针),暂不修改其它特征.再把前一行中除了第一段之外的所有其它与 THIS 邻接的、标号为 p_2 的行程的特征指针,修改为 THIS 的特征指针.然后在等价数组中记录标号等价关系.之后合并、修改连通域特征.最后修改前一行中第一段与 THIS 邻接的、标号为 p_2 的行程的特征指针,为 THIS 的特征指针.

经过上述操作,标号和其它特征通过 THIS 的特征指针的传递而得到逆向传播.

4　实　验

由于特征在标号过程中或在标号完成后计算在方法或时间上没有本质区别,为与其它连通域标号算法进行比较,这里仅考虑 FCCA 算法的标号部分,即特征数据块中仅包含标号特征.

用来对比的有三种著名的算法:一是 A. Rosenfeld 和 P. Pfaltz 提出的"经典算法"[3];一是 R. M. Haralick提出的"循环算法"[4];还有一种由 R. Lumia、L. Shapiro 和 O. Zuniga 提出的"Lumia 算法"[2].这些算法提出时所考虑的输入都是二值图像,它们对图像中的每一个像素标号.由于 FCCA 算法是对目标像素的行程标号,而一般认为对像素行程进行标号本身就应比对每个像素进行标号要快得多[1],所以为了比较的公平性,在实验中已将这三种著名算法全部编写为相应的以行程为标号对象的程序,并且下面所给出的结果都没有考虑图像行程编码和解码时间(事实上,行程编解码占用的时间与标号所需时间相比极少).显然,此时的算法运行时间是与图像中所含有的行程数和连通域数有关而不是与像素数有关.

下面先以经典算法为例,给出它的以行程为标号对象的算法描述.循环算法和 Lumia 算法的相应行程算法描述可类似地给出,文中不再赘述.

经典算法(行程改写版):

第一遍扫描:

STEP 1　创建一个空的等价表.

STEP 2　对每行每个行程段,按以下规则分配标号并记录等价关系:

①找到当前行程段在前一行中的相邻行程段集合.

②如果该集合为空,则给当前行程段分配一个新的标号;否则,取该集合行程段的标号中最小的一个分配给当前行程段.

③如果该集合中的行程段有不同的标号,把它们加入等价表.

第二遍扫描:

STEP 3　根据等价表建立等价类表.

STEP 4　对已标过号的每行每个行程段:寻找该行程段的标号所在等价类中的最小标号,把找到的最小标号重新赋给当前行程段. ◇

实验采用了 5 幅图像及其衍生的图像,共 10 幅.5 幅源图像中,3 幅较小的为人工合成的,另 2 幅是将同一个 pdf 文本在屏幕上不同显示比例下的图像二值化所得(分别相当于800×600≈469K和2 400×1 800≈4 219K 的图像).将 2 幅实际图像的目标和背景对调,得到 2 幅新图像;另将合成和实际图像各选 1 幅比例放大,得 3 幅新图像.表 1 中列出了 10 幅实验图像中所含的行程数和连通域数.分别使用 P3/450, 64M RAM 的 PC 机和 P3/600, 128MRAM 的笔记本电脑运行 4 种算法,得到表 2 至表 5 的结果.由于是在 Win98 环境下运行程序,有许多后台任务不时占用 CPU,所以实验采用了在程序中将每种算法本身循环 500 次、再将程序运行 100 遍、然后求其时间平均作为计算算法每次运行所需时间的策略,以期将后台任务对时间统计的不均匀性降低.

表1 图像及其中所含行程数和连通域数

图像代号	1	2	3	4	5	6	7	8	9	10
文件名	ttt	spiral	bigspiral	manycc	page	page_i	p_2	p_3	page3x	page3x_i
行程数	71	2716	5432	3796	20212	21012	40424	60636	68492	70442
连通域数	1	2	2	200	1838	1171	1838	1838	2869	686

表2 使用台式计算机,标号为非连续自然数(时间单位:毫秒)

图像代号	1	2	3	4	5	6	7	8	9	10
FCCA	0.37	0.73	1.10	0.89	5.55	5.94	10.66	15.17	18.45	20.29
经典	1.11	2.06	2.92	2.41	10.36	10.50	19.87	28.72	32.90	34.09
循环	0.13	7.78	14.93	4.53	54.25	30.37	113.81	178.06	180.58	117.75
Lumia	0.46	2.09	3.70	4.35	82.71	74.11	147.44	212.81	378.78	212.42

表3 使用台式计算机,标号为连续自然数(时间单位:毫秒)

图像代号	1	2	3	4	5	6	7	8	9	10
FCCA	0.78	1.32	1.91	1.69	8.89	9.54	17.67	25.51	30.19	32.34
经典	2.32	3.46	4.52	3.90	14.25	14.41	27.51	39.83	45.30	46.61
循环	1.34	9.18	16.59	6.01	56.78	33.56	122.13	191.12	195.55	133.06
Lumia	1.67	3.52	5.44	5.87	86.30	77.85	155.06	223.90	390.99	224.76

表4 使用笔记本电脑,标号为非连续自然数(时间单位:毫秒)

图像代号	1	2	3	4	5	6	7	8	9	10
FCCA	0.14	0.45	0.88	0.55	6.01	6.15	10.60	14.41	16.88	18.35
经典	1.58	2.42	3.32	2.76	10.38	10.87	17.26	25.48	29.04	30.01
循环	0.11	5.38	10.13	3.12	46.32	26.34	100.83	153.91	153.71	102.03
Lumia	0.19	1.41	2.70	3.00	63.67	57.38	114.26	164.21	286.77	164.71

表5 使用笔记本电脑,标号为连续自然数(时间单位:毫秒)

图像代号	1	2	3	4	5	6	7	8	9	10
FCCA	0.53	1.33	2.23	1.66	9.71	9.72	17.06	22.89	26.43	28.21
经典	2.45	3.58	4.92	4.08	14.85	15.47	24.73	36.50	41.22	42.41
循环	0.83	6.20	11.33	4.11	51.52	31.04	108.58	167.30	168.49	119.17
Lumia	0.84	2.14	3.87	4.20	69.21	62.37	121.10	174.93	298.07	177.80

从表中可以明显地看到FCCA算法的高效性.在连通域分析的重点是特征计算而不关心标号的结果时FCCA算法更显其优越性.这是因为FCCA算法只使用了一遍扫描,并且没有多余的循环和行内扫描,所以它比需要重新标号扫描的经典算法、需要额外循环的循环算法、以及需要逆向传播标号的Lumia算法都要快许多.表中的结果与文献[14]的结果一样说明了Lumia算法在大内存的计算机上没有明显的优势.另外,当采用对行程标号的方法时,经典算法反而显示出生命力.

若要求区域标号是连续自然数,即最大的区域标号等于图像中连通域的数目,那么仅有经典算法和本文算法可通过全局等价表而不需额外的图像扫描达到这一要求;而循环算法和Lumia算法,或者需要额外的图像扫描或者需要额外的等价表来把不连续的标号转换为连续的标号.

从上述的分析可知,本文的算法是一种很有吸引力的连通域分析算法.

参 考 文 献

[1] Rosenfeld A, KaK A C, 著; 李叔梁,等译. 数字图像处理. 北京: 科学出版社, 1983

[2] Lumia R, Shapiro L, Zuniga O. A New Connected Components Algorithm for Virtual Memory Computers. Computer Vision, Graphics, and Image Processing, 1983, 22: 287 - 300

[3] Rosenfeld A, Pfaltz P. Sequential Operations in Digital Picture Pro-

cessing. Journal of the Association for Computing Machinery, 1966, 12: 471 - 494

[4] Haralick R M. Some Neighborhood Operations. In: Onoe M, Preston K, Rosenfeld A, eds. Real Time/Parallel Computing Image Analysis, Plenum Press, New York, 1981

[5] Dinstein I, Yen D W L, Flickner M D. Handling Memory Overflow in Connected Component Labeling Applications. IEEE Trans on Pattern Analysis and Machine Intelligence, 1985, 7(1): 116 - 121

[6] Han Y J, Wagner R A. An Efficient and Fast Parallel - Connected Component Algorithm. Journal of the Association for Computing Machinery, 1990, 37(3): 626 - 642

[7] Yang X D. An Improved Algorithm for Labeling Connected Components of a Binary Image. In: Rosenfeld A, Shapiro L, eds. International Journal of Computer Vision, Graphics, and Image Processing, Special Volume: Progress in Computer Vision and Image Processing, Academic Press, 1992, 555 - 569

[8] Choudhary A, Thakur R. Connected Component Labeling on Coarse Grain Parallel Computers: An Experimental Study. Journal of Parallel and Distributed Computing, 1994, 20(1): 78 - 83

[9] Suzuki K, Horiba I, Sugie N. Fast Connected-Component Labeling through Sequential Local Operations in the Course of Forward Raster Scan Followed by Backward Raster Scan. Trans of Information Processing Society of Japan, 2000, 41(11): 3070 - 3081

[10] Kistler J J, Webb J A. Connected Components with Split and Merge. In: Proc of the 5th International Parallel Processing Symposium, Anaheim, CA, 1991, 194 - 201

[11] Choudhary A, Thakur R. Evaluation of Connected Component Labeling Algorithms on Shared and Distributed Memory Multiprocessors. In: Proc of the 6th International Parallel Processing Symposium, Beverly Hills, CA, 1992, 362 - 365

[12] Greiner J. A Comparison of Parallel Algorithms for Connected Components. In: Proc of the 6th Annual ACM Symposium on Parallel Algorithms and Architectures, Cape May, New Jersey, 1994, 16 - 23

[13] Bulgarelli A, Stefano L D. A Simple and Effcient Connected Component Labeling Algorithm. In: Proc of the 10th International Conference on Image Analysis and Processing, Venice, Italy, 1999, 322 - 327

[14] Park J M, Looney C G, Chen H C. Fast Connected Component Labeling Algorithm Using a Divide and Conquer Technique. In: Proc of the ISCA 15th International Conference on Computers and Their Applications, New Orleans, LA, 2000(Also available, //cs.ua.edu/TechnicalReports/TR - 2000 - 04.pdf)

A FAST CONNECTED COMPONENT ANALYSIS ALGORITHM AND ITS IMPLEMENTATION

Kong Bin

(*Department of Automation, University of Science and Technology of China, Hefei* 230027)

(*Hefei Institute of Intelligent Machines, Chinese Academy of Sciences, Hefei* 230031)

ABSTRACT

With current PC configuration, the memory requirements for connected components labeling algorithms are easily satisfied while the time costs usually still cannot meet practical demand. Here a fast algorithm calculating shape/grayscale features of connected components while labeling them is proposed. It is executed on pixel runs and labels are considered as one of the features, which associate to runs or objects through a data pointer. A chain mechanism is designed to represent and realize the equivalence of labels. By passing the pointers and dynamically modifying the feature data, equivalence and back propagation of labels are realized while the label and other features are passed on. The later two techniques make connected components anslysis can even be implemented through only one image scanning. Experimental results show the effectiveness of the algorithm.

Key Words Connected Component Analysis (CCA), Connected Component Labeling (CCL), Run Length, Chain Mechanism for Equivalence, Pointer