单次扫描连通域分析算法研究综述

曲立国[1,2],陈国豪[1],胡 俊[1],陈 鹏[1]
(1. 安徽师范大学物理与电子信息学院,安徽芜湖 241002; 2. 安徽省智能机器人信息融合与控制实验室,安徽芜湖 241002)

摘 要: 连通域分析在单次扫描中标记像素同时提取每个连通域的特征数据,是二值图像处理的重要步骤之一. 基于FPGA的硬件架构实现单次扫描连通域分析算法可以实现快速位流图像实时处理. 本文重点分析了近十年来发展的连通域标记算法和单次扫描连通域分析算法,阐述了典型连通域分析算法的实现策略和框架,给出了它们的伪代码,并描述了它们的联合查找算法. 此外,通过数据对比,从算法硬件架构的内存需求和吞吐量等方面对不同算法的性能进行了比较分析,并总结了它们的优缺点. 分析结论为实现基于FPGA高速位流图像的连通域检测提供了理论依据和数据参考.

关键词: 连通域分析; 连通域标记; 特征提取; 图像处理; FPGA
中图分类号: TP211 **文献标识码:** A **文章编号:** 0372-2112(2022)06-1521-16
电子学报URL:http://www.ejournal.org.cn **DOI:**10.12263/DZXB.20210368

A Review of Single-pass Connected Component Analysis Algorithms

QU Li-gou[1,2], CHEN Gou-hao[1], HU Jun[1], CHEN Peng[1]
(1. *School of Physics and Electronic Information, Anhui Normal University, Wuhu, Anhui* 241002, *China*;
2. *Anhui Provincial Engineering Laboratory on Information Fusion and Control of Intelligent Robot, Wuhu, Anhui* 241002, *China*)

Abstract: Connected component analysis(CCA) is one of the major steps in binary image processing, which label pixels in single-pass and extract the features of each connected component at the same time. Single-pass connected component analysis algorithm based on FPGA hardware architecture can realize fast real-time bit stream image processing. In this article, we focus on connected component labeling(CCL) algorithms and single-pass connected component analysis algorithms developed in the last decade, explain the implementation strategies and architectures of the typical connected component analysis algorithms, present their pseudo codes, and describe their Union-find algorithms. In addition, through data comparison, the performance of different algorithms is compared and analyzed in terms of memory requirements and throughput of algorithm hardware architectures, and their advantages and disadvantages are summarized. The analysis results provide a theoretical basis and data reference for the realization of connected component detection based on FPGA high-speed bit stream images.

Key words: connected component analysis; connected component labeling; feature extraction; image processing; FPGA

1 引言

连通域标记(Connected Component Labeling, CCL)和基于CCL改进的连通域分析(Connected Component Analysis, CCA)是二值图像分析和图像处理的重要步骤,在医学[1~3]、自动监控系统[4~6]、天气监测[7]、人脸识别[8]和天文观测[9]等诸多领域有广泛应用. 它对二值图像中前景像素(也称对象像素)进行标记,让每个单独的连通域形成一个被标识的块,进而获取这些区域的面积、边界框和质心等相关图像特征信息.

为了不失一般性,在$W\times H$像素大小的二值图像中,坐标为(x,y)的像素表示为$b(x,y)$,其中$0\leqslant x\leqslant W-1$,$0\leqslant y\leqslant H-1$. 图像内像素值只有0和1,假设0代表背景像素,1代表前景像素.

CCL算法常见的连通关系有四连通和八连通两种,如图1所示. 任意像素$b(x,y)$的上下左右四个像素,即$b(x,y\text{-}1)$,$b(x,y+1)$,$b(x-1,y)$,$b(x+1,y)$被称为

收稿日期:2021-03-16;修回日期:2022-03-09;责任编辑:王天慧
基金项目:安徽省高等学校自然科学研究项目重点项目(No.KJ2019A0511)

它的四连通像素. 四连通像素加上四个对角相邻像素，即$b(x-1,y-1)$,$b(x+1,y-1)$,$b(x-1,y+1)$,$b(x+1,y+1)$被称为它的八连通像素.

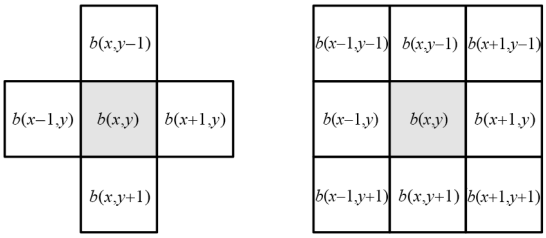


图1 四连通和八连通

根据图1的连通关系，相互连通的对象像素属于同一个连通域. 连通域标记就是给属于一个连通域的所有对象像素分配相同标签，利用唯一的标签区分图像中不同连通域. 连通域是一副图像中相互连通的对象像素的最大集合，所以一个连通域也可以被称为一个物体对象.

自20世纪60年代以来，大量学者致力于连通域标记算法的研究. Rosenfeld等人[10]和Lumia等人[11]设计了基于像素的连通域标记算法；Ronse等人[12]设计了基于游程的连通域标记算法. 现代算法都是在它们的基础上发展而来的，根据算法通过图像的扫描次数的不同，其可以分为以下几类：(1)多次扫描标记算法[13,14]；(2)基于轮廓跟踪的标记算法[15~18]；(3)两次扫描标记算法[19~39]；(4)单次扫描标记算法[40~68].

已有一些文章对CCL算法进行了比较. 文献[30]对文献[10,11,15,25~28]中提出的算法进行了综述，并提出了用于测试不同算法性能的基准. 此外，文献[35]对基于标签传播[15]和基于标签等价解析[22~26,32]的标记算法进行了综述，详细介绍了一些典型算法的原理，并给出了一些算法的伪代码. 但这些比较综述主要针对两次扫描CCL算法，很少提及单次扫描算法.

单次扫描连通域标记算法又称为单次扫描连通域分析(CCA)，也有文献对CCA算法进行了比较. 文献[50]对一些现有CCL和CCA算法进行了比较，然而这篇文章对除了文献[48]算法以外的所有算法都只是进行了简单的描述，并没有详细说明算法的原理和算法之间的联系. 为此本文对最近十年来发展的单次扫描CCA算法进行分析，描述适用于单次扫描CCA算法的联合查找算法，阐述典型CCA算法的原理，给出它们的伪代码，并对算法FPGA性能的实现情况进行了比较.

2 联合查找算法

无论是在CCL算法还是CCA算法中，标签等价关系的记录和处理都是算法实现的重要操作，并且影响着算法处理效率. 联合查找(Union-find)算法是记录和解析标签等价关系的一种有效方法[50]，它通过一个联合查找森林结构实现，森林由相互独立的联合查找树组成. 联合查找树的特点：(1)给每个连通域分配的临时标签都是树中的一个结点，每个结点都连接到它所在树中的父结点；(2)树中的所有结点都是等价的，一棵树代表一个连通域；(3)如果树中一个结点的父结点就是它本身，那么该结点被称为该树的根结点，根结点是树中最小的标签，对应着该树代表连通域的根标签.

可以采用图像进一步表示联合查找树算法的结构，做以下定义：结点N_i通过有向边指向另一个结点N_j，有向边E可以表示为$N_i \rightarrow N_j$，其中N_i是N_j的子结点，N_j是N_i的父结点. 一个结点可以有很多子结点，没有子结点的结点是叶结点. 所有结点的集合$N(F)$和有向边集合$E(F)$构成森林F，如式(1)所示：

$$\begin{aligned} &F=(N,E)\\ &N(F)=\left\{N_0,N_1,\cdots N_{n-1}\right\}\\ &E(F)=\left\{\left(N_{i_0}\rightarrow N_{j_0}\right),\left(N_{i_1}\rightarrow N_{j_1}\right),\cdots,\left(N_{i_{m-1}}\rightarrow N_{j_{m-1}}\right)\right\} \end{aligned} \tag{1}$$

每一个结点都沿着一条路径连接到树的根结点，路径中有向边E的条数就是该结点的等级. 根结点的等级为0，其它结点比它的父结点等级高1，叶结点的等级最高，树中叶结点的等级就是这棵树的高度. 两个连通域和它们对应的联合查找树结构F如图2所示.

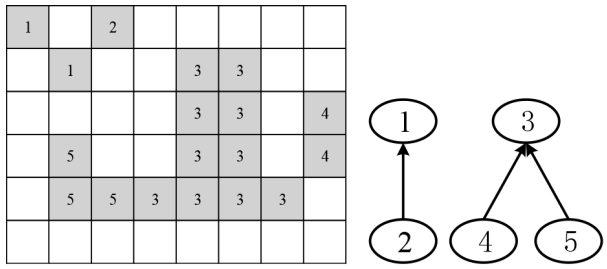


图2 连通域示例和联合查找树数据结构F

联合查找树已经衍生出多种改进算法，如快速查找(QuickFind)算法[37]、快速联合(QuickUnion)算法[37]和带有路径压缩的快速联合(QuickUnion with Path Compassion)算法[37]. 在单次扫描CCA算法中，一种有效的联合查找算法是文献[50]中提出的基于上下文(Context-based)的联合查找算法，它不仅结合了快速查找和快速联合的最佳特性，还加入年龄平衡(Age-balancing)[39]策略. 年龄平衡策略不再依据标签数值大小定义标签大小，而是依据分配标签的时间先后顺序(即标签“年龄”)，越早分配的标签年龄越大，标签越小. 年龄平衡策略可以确保给连通域分配的初始标签始终是根标签，避免许多复杂的合并，简化了算法计算复杂度.

基于上下文的联合查找算法伪代码如算法1所示,它主要包含新树(Newtree)、查找(Find)、联合(Union)和展平(Flatten)四个操作. 当给新连通域分配新标签时,将会在森林中创建一个新树,树的根结点n指向自己. 通常,查找操作会迭代找出结点的根结点[38],但算法1中查找操作只找出结点n的父结点. 在联合操作中连接结点u和v,需要对两个结点进行一次查找,找出它们的父结点e和f. 当e小于f时,把e作为f的父结点;而当e大于f时,不仅要把f作为e的父结点,还需要把这两个合并标签压入链堆栈中保存. 展平操作在图像每一行行尾执行,将链堆栈中的标签反向弹出,把较小标签的父结点作为较大标签的父结点,从而解析两个标签的等价关系.

算法1　基于上下文的联合查找算法

```
1   Newtree(node n)
2       T[n]=n
3   END OF Newtree
4   Find(node n)
5       RETURN T[n]
6   END OF Find
7   Union(node u, node v)
8       e=Find(node u)
9       f=Find(node v)
10      IF e < f  THEN
11          T[f]=e
12      ELSE
13          Stack.Push(f, e)
14          T[e]=f
15      END IF
16  END OF Union
17  Flatten
18      WHILE !Stack.empty DO
19          L_min, L_max=Stack.pop
20          T[L_max]=Find[L_min]
21      END WHILE
22  END OF Flatten
```

3　连通域标记

表1从以下几个方面对典型CCL算法和CCA算法的性能进行了比较:(1)扫描次数和连通性;(2)扫描和标记单元;(3)运行时间复杂度和采用的等价解析策略;(4)是否重用标签. 图3列举了CCL算法和CCA算法主要的扫描窗口,其中图3(d)是文献[54]中提出的TRSP(Tang's Run-based Single Pass)算法的扫描窗口,该算法将会在4.2.1节分析.

Rosenfeld等人[10]在1966年提出的OCL(Original Connected-component Labeling)算法是经典连通域标记算法,它采用如图3(a)所示的扫描窗口对输入图像进行两次扫描并生成一个包含标记结果的输出图像. 第一次扫描图像时给像素分配临时标签:当前像素如果是背景像素,则标记为0;如果是对象像素,需要根据四个邻域像素信息为当前像素选择临时标签. 如果邻域都是背景像素,则分配一个新标签;如果邻域内只存在一个标签,则把该标签分配给当前像素;如果邻域中存在两个不同标签,则把较小标签分配给当前像素,并在等价表中记录两个标签的等价关系. 第二次扫描图像

表1　典型CCL和CCA算法的性能比较

算法缩写	形式	扫描单元	标记单元	连通性	运行时间复杂度	等价关系解析策略	是否标签重用
OCL[10]	Two-pass CCL	Pixel	Pixel	8	—	Rosenfeld	—
SCL[14]	Multi-pass CCL	Pixel	Pixel	8	Linear	Iterative connection table	—
CT[15]	Two-pass CCL	Pixel	Pixel	8	Linear	None	—
CTCL[22]	Two-pass CCL	Pixel	Pixel	8	Linear	Equivalent label set	—
ICTCL[23]	Two-pass CCL	Pixel	Pixel	8	Linear	Equivalent label set	—
RCL[24]	Two-pass CCL	Run	Run	8	Linear	Equivalent label set	—
BCL[26]	Two-pass CCL	Block	Block	8	Linear	QuickUnion+path compression	—
OSP[42]	Single-pass CCA	Pixel	Pixel	8	Linear	Context-based union-find	No
IOSP[45]	Single-pass CCA	Pixel	Pixel	8	Linear	Context-based+relabeling	Yes
SLSP[48]	Single-pass CCA	Pixel	Pixel	8	Linear	Context-based union-find	Yes
DLSP[50]	Single-pass CCA	Pixel	Run	8	Linear	Context-based union-find	Yes
ZZSP[51]	Single-pass CCA	Pixel	Run	8	Linear	Context-based union-find	Yes
TRLE[52]	Single-pass CCA	Run	Run	8	—	QuickUnion	Yes
ZRSP[53]	Single-pass CCA	Pixel	Run	4	Linear	Multi-layer-index structure	Yes
TRSP[54]	Single-pass CCA	Run	Run	4	Linear	Linked list	Yes
CSP[56]	Single-pass CCA	Cell	Cell	4	Linear	Relabeling memory	Yes
JSP[58]	Single-pass CCA	Pixel	Pixel	8	Linear	None	Yes

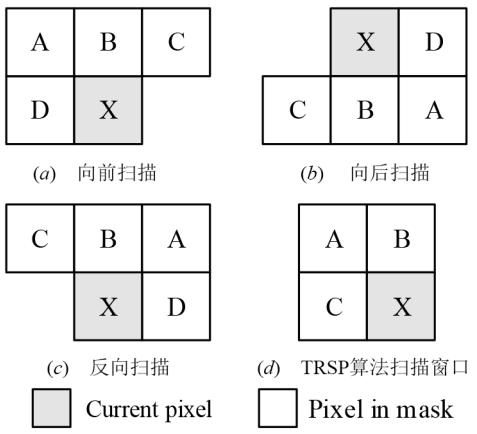


图3 CCL和CCA算法的扫描窗口

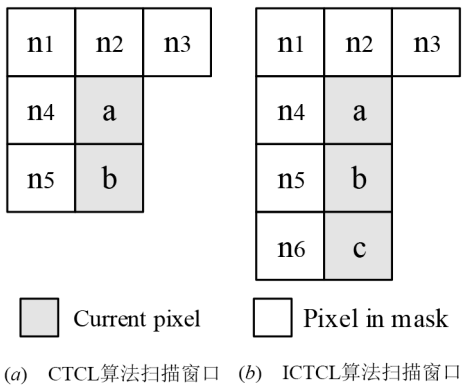


图4 CTCL和ICTCL算法的扫描窗口

时用等价表中的代表标签(连通域中最小的标签)替换所有临时标签. OCL算法需要为最终图像和等价表分配足够的内存,并且由于在检测等价关系时重复采用排序算法,计算量较大.

为了减少内存需求,Haralick等人[13]提出一种多次扫描算法,该算法根据邻域像素信息对二值图像进行交替地向前扫描(图3(*a*))和向后扫描(图3(*b*)),在交替扫描中处理标签等价关系,不需要等价表和额外内存空间. 交替扫描图像使得算法扫描次数不固定,连通域几何形状越复杂,扫描次数越多. Suzuki等人[14]提出的SCL(Suzuki's Connected-component Labeling)算法对Haralick等人的算法做出了改进,采用标签连接表记录和处理标签等价关系. SCL算法也需要多次扫描,作者证明最大扫描次数为4次,算法执行时间与图像连通域中的像素数成正比.

Chang等人[15]提出的算法采用了完全不同的方法,通过轮廓跟踪(Contour Tracing,CT)技术标记连通域. CT算法从上到下、从左到右逐行扫描二值图像,当遇到一个连通域的外部轮廓或内部轮廓时,采用轮廓跟踪技术给轮廓上所有对象像素分配相同的标签,执行轮廓跟踪的同时也标记与轮廓相连的背景像素以保证轮廓只会被跟踪一次. 标记完连通域外部轮廓之后,整个连通域的标签就确定了,给连通域内部像素分配的标签都与外部轮廓标签相同. 作者已经证明CT算法时间复杂度与图像大小呈线性关系.

He等人[22]提出的基于配置转换的标记算法(Configuration Transition-based Connected component Labeling, CTCL)采用如图4(*a*)所示的扫描窗口一次性处理两个像素,通过考虑扫描窗口中像素的9种配置的转换,尽可能多地利用处理后两个像素时检测到的信息处理当前两个像素,使在第一次扫描中要检查的像素平均数目减少,从而加快算法处理速度. Zhao等人[23]提出的ICTCL(Improved CTCL)算法对CTCL算法做出改进,进一步减少处理像素要检查的像素平均数目,采用如图4(*b*)所示的扫描窗口一次处理三个像素,考虑的配置有16种. CTCL算法和ICTCL算法都采用等价标签集(Equivalent Label Set)策略记录和解析标签等价关系. 文献[35]已经证明这两种算法的时间复杂度都与图像大小呈线性关系. 本文不介绍等价标签集策略,相关细节参考文献[22,23].

He等人[24]提出的基于游程的一次半扫描算法(Run-based Connected-component Labeling,RCL)采用等价标签集策略记录和解析标签等价关系. RCL算法把一个游程视为一个超级像素,在第一次扫描中给每个游程而不是给游程中的像素分配临时标签. 由于第一次扫描记录了所有游程数据,第二次扫描只需重新扫描对象像素而不用扫描背景像素,故作者称算法为一次半扫描算法. 一般图片中游程数量比对象像素数量少得多,采用基于游程的技术将大大减少分配标签的成本. RCL算法时间复杂度与图像大小呈线性关系,虽然作者声称该算法是一次半扫描,但是实质上还是两次扫描算法.

Grana等人[26]提出的基于像素块的两次扫描标记算法(Block-based Connected-component Labeling, BCL)采用等价标签集策略记录和解析等价标签. 由于一个2×2的块内所有像素必定属于同一连通域,BCL算法把基于像素的标签处理扩展到块上,将如图3(*a*)所示的扫描窗口中的像素替换成2×2的块,扩展后可以一次实现对20个像素的连通关系分析. BCL算法在第一次扫描中给像素块分配临时标签并记录块之间的等价关系,在第二次扫描中把块的代表标签分配给块中每个对象像素. BCL算法时间复杂度与图像大小呈线性关系.

4 连通域分析

CCL在处理图像时需要两次扫描,第一次扫描分配临时标签,第二次扫描重新标记,并生成一个和图像

同样大的标记图像. 图像分析中最重要的是连通域的特征数据提取,标记图像本质上只是一个辅助图像数据结构. 如果在第一次扫描中直接提取特征数据,那么第二次扫描就可以省去,两次扫描就可以减少到单次扫描. 在单次扫描图像的同时提取特征数据并动态地解决数据关联问题,这是近十年来研究发展的单次扫描CCA算法的原理. CCA算法可以处理位流图像,且不需要储存标签图像,只需要少量内存,这些特性使它们适合在FPGA上实现,完成图像实时高速处理,因此单次扫描CCA算法都是基于FPGA硬件架构设计的.

4.1　基于像素的单次扫描CCA算法

基于像素的单次扫描CCA算法处理图像时考虑像素之间的连接关系,本节分析以下几种CCA算法:

(1)原始的单次扫描(Original Single Pass, OSP)CCA算法[42];

(2)改进的OSP(Improved OSP, IOSP)CCA算法[45];

(3)单次查找单次扫描(Single Lookup Single Pass, SLSP)CCA算法[48];

(4)双重查找单次扫描(Double Lookup Single Pass, DLSP)CCA算法[50];

(5)"Z"字形单次扫描(Zig-Zag Single Pass, ZZSP)CCA算法[51].

4.1.1　OSP算法

Bailey等人[42]提出的OSP算法采用基于上下文的联合查找算法记录和解析标签等价关系,它并不回收标签,分配标签的顺序按照标签数值由小到大排序,因此不引入年龄平衡策略. OSP算法的硬件架构如图5所示,主要包含邻域标签处理模块、标签选择模块、合并控制和合并表模块以及数据表模块,行缓冲区只缓存一行图像像素.

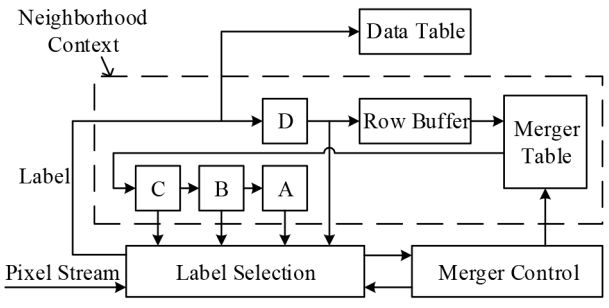


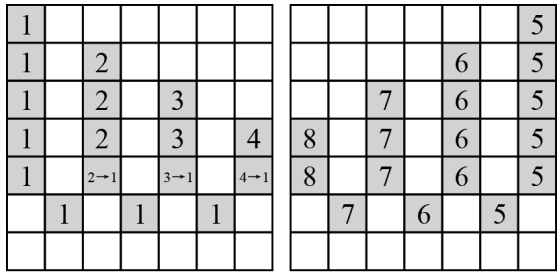
图5　OSP算法的架构

OSP算法的扫描窗口和标签选择策略与OCL算法相同,由邻域标签处理模块给扫描窗口提供邻域标签L_A,L_B,L_C和L_D,标签选择模块根据邻域标签信息给当前像素分配标签L_X,同时数据表累加连通域的特征数据. 为了减少对内存访问次数,邻域标签更新采用标签传播方法,执行步骤如算法2所示,其中添加符号"′"的标签表示上一时钟周期的标签.

算法2　标签传播

```
1  L_A=L_B′
2  L_D=L_X′
3  IF B & D THEN
4    L_B=L_X′
5  ELSE
6    L_B=L_C′
7  END IF
8  L_C=Find(C)
```

当发生标签合并时,合并控制器控制合并表记录合并操作中生成的标签等价关系. 如图6(*a*)所示的传播合并模式生成的标签等价关系可以直接记录在合并表中,因为较大标签在合并之后不再出现(链的长度总是为1),不会影响后续标记结果. 而如图6(*b*)所示的非传播合并模式会形成一条长标签链,被称为链式效应[54]. 对于非传播合并模式,必须把两个合并标签记录在一个链堆栈中,在行尾展平操作中解析标签等价关系该操作被(称为解链).



(*a*)　传播合并模式　　(*b*)　非传播合并模式

图6　两种合并模式

OSP算法时间复杂度与图像大小呈线性关系,数据表和合并表大小与标签数量成正比,储存合并标签的链堆栈大小与图像宽度W成正比.

4.1.2　SLSP算法

OSP算法的标签数量很多,与标签数量相关的结构占用了大量内存空间,为了减少内存需求,回收标签方法被提出. Ma等人[45]发现一行像素中标签数量最多为图像宽度的一半,把标签数量压缩到这样大小可以节省可观的内存. Ma在IOSP算法中采用了激进的重标记方法实现了这一点,在图像每一行开头采用标签0重新标记图像,使得标签数目最大为$W/2$,这种重标记技术需要一个转换表转换两行标签的连接关系.

Klaiber等人[48]提出的SLSP算法采用一种新的标签回收方法改进了OSP算法,基于硬件实现的算法如

算法3所示，对每个像素执行标签传播（Label Propagate）、更新数据结构（Update Data Structure）和解析过期标签（Resolve Stale Labels）三个操作，在一行像素行尾执行展平操作. 采用标签回收策略后，更新数据结构具体操作如算法4所示.

算法3 SLSP算法

```
FOR y=0 TO H-1 DO
  FOR x=0 TO W-1 DO
    Label Propagate
    Update Data Structure
    Resolve Stale Labels
  END FOR
  Flatten
END FOR
```

算法4 更新数据结构

```
IF X THEN
  IF !A & !B & !C & !D THEN                    △New label
    L_X=Newtree
    V[L_X]=TRUE
  ELSE                                         △label merge
    IF (A | D) & C & (L_{A or D} ≠ L_C) THEN
      L_X=Union(L_{A or D}, L_C)
      IF V[Max(L_{A or D}, L_C)] THEN
        Max(L_{A or D}, L_C) to reuse
      END IF
      V[Max(L_{A or D}, L_C)]=FALSE
    ELSE                                       △label copy
      L_X=LabelCopy(L_{A or D}, L_B, L_C)
    END IF
    IF !V[L_X] THEN
      Push(L_X) to SLS
    END IF
  END IF
  AT[L_X]=Y
ELSE
  L_X=0
END IF
```

当连通域的所有像素都被标签标记，我们称连通域标记完成，标记标签在后续扫描中不会再出现，这种标签可以回收重利用. SLSP算法的标签回收策略采用活动标签AT（Active Tag）关联每个标签，在每一行动态地检查连通域是否标记完成，通过回收标签减少标签数量. 重用标签打乱了标签排列顺序，因此算法中引入增强标签（Augmented label）实现年龄平衡策略，增强标签由行号$L\cdot\mathrm{rw}$与索引$L\cdot\mathrm{index}$组成，如图6所示的两种合并模式可以通过式（2）评估，其中$L_{A\text{ or }D}$表示L_A或L_D.

$$\begin{aligned} L_{A\text{ or }D}<L_C \quad & \text{when } L_{A\text{ or }D}\cdot\mathrm{rw}\leqslant L_C\cdot\mathrm{rw} \\ L_{A\text{ or }D}>L_C \quad & \text{when } L_{A\text{ or }D}\cdot\mathrm{rw}>L_C\cdot\mathrm{rw} \end{aligned} \tag{2}$$

OSP算法在行尾解链使联合查找树的高度不大于1，但在下一行中高度为1的树和另一棵树合并将有可能产生高度为2的树. 在行尾之前，如果再次遇到树的叶结点（等级为2），进行一次查找将无法访问到树的根标签，这种叶结点对应的标签称为过期标签. 算法设置一个有效性标记V（Valid tag）跟踪标签是否为过期标签，过期标签将记录在过期标签堆栈SLS（Stale Label Stack）中，随后在解析过期标签操作中处理. 当过期标签被分配给当前像素时，特征数据将无法累加到正确的标签上，需要先把特征数据暂存起来，等过期标签的根标签进入邻域时再将特征数据累加到根标签上.

SLSP算法的时间复杂度与图像大小呈线性关系，数据表、合并表和链堆栈大小都与图像宽度W成正比.

4.1.3 DLSP算法

处理过期标签需要额外的操作步骤，这导致算法复杂度变高，处理速度变慢. Klaiber等人[50]提出DLSP算法，采用双重查找消除了过期标签，并证明双重查找一定会找到根标签，这样消除了和过期标签相关的任何操作，降低了算法复杂度，算法实现如算法5所示.

DLSP算法以像素为扫描单元，以游程为标记单元，把基于像素的处理与基于游程的处理统一起来，进一步降低了算法计算复杂度. 同一游程的所有像素都有相同根标签，因此只需要对游程中第一个像素进行双重查找就可以找到整个游程的根标签，后续像素标签只需要传递而无需再次查找.

算法5 DLSP算法

```
FOR y=0 TO H-1 DO
  FOR x=0 TO W-1 DO
    Label Propagate
    Update Data Structure
  END FOR
  Flatten
END FOR
```

DLSP算法时间复杂度与图像大小呈线性关系，在存储空间上与SLSP算法相比减少了过期标签相关的数据结构.

4.1.4 ZZSP算法

DLSP算法仍然存在链式效应，需要消耗额外的时钟解析标签链，Bailey等人[51]提出的ZZSP算法解决了这个问题. ZZSP算法采用了“Z”字形（Zig-Zag）扫描替换光栅式扫描，交替采用如图3（a）和图3（c）所示的扫描窗口每隔一行以相反方向扫描，在下一行扫描中动态解析链式效应，也消除了行尾的展平操作. ZZSP算法实现如算法6所示，对每个像素只需执行标签传播

(Label Propagate)和更新数据结构(Update Data Structure)两个操作. 当扫描从行尾进入到下一行行首时,需要翻转窗口以调整扫描顺序,操作步骤如算法6中第5~7行所示.

算法6　ZZSP算法

```
1    Line=FALSE
2    FOR y=0 TO H−1 DO
3        FOR x=0 TO W−1 IF y=even ELSE x=W−1 to 0 DO
4        IF Line THEN
5            L_{A or D}=0
6            L_B=L_X
7            L_C=L_D
8            Line=FALSE
9        ELSE
10           Label Propagate
11       END IF
12       Updata Data Structure
13       END FOR
14       Line=TRUE
15   END FOR
```

与年龄平衡策略类似,ZZSP算法依据扫描顺序定义两个像素之间的优先级关系,如式(3)所示:

$$\begin{aligned}
&P_1<P_2 \quad \text{when } P_1\cdot y<P_2\cdot y\\
&P_1<P_2 \quad \text{when } \left(P_1\cdot y=P_2\cdot y\right)\wedge\left(P_1\cdot y \text{ is even}\right)\wedge\\
&\qquad\qquad \left(P_1\cdot x<P_2\cdot x\right)\\
&P_1<P_2 \quad \text{when } \left(P_1\cdot y=P_2\cdot y\right)\wedge\left(P_1\cdot y \text{ is odd}\right)\wedge\\
&\qquad\qquad \left(P_1\cdot x>P_2\cdot x\right)\\
&P_1>P_2 \quad \text{otherwise}
\end{aligned} \tag{3}$$

其中$P_1<P_2$表示P_1优先级更高.

在SLSP和DLSP算法中,活动标签AT只记录与标签相关的行号,而ZZSP算法在数据表DT(Data Table)中引入AT,记录与每个标签相关的坐标,如式(4)所示:

$$DT(L_X)\cdot \mathrm{AT}\begin{cases}(y+1,x-1), & \text{when } y \text{ is even}\\ (y+1,x+1), & \text{when } y \text{ is odd}\end{cases} \tag{4}$$

当扫描窗口通过该坐标,即确定该标签对应连通域已经标记完成时,标签将回收重利用. 在标签复制和合并操作中,活动标签会跟随数据表一起更新,活动标签的更新将遵循公式(3)的优先级定义,对于两个活动标签AT_1和AT_2,当AT_1的优先级高于AT_2时,将会保留AT_2,反之保留AT_1.

ZZSP算法时间复杂度与图像大小呈线性关系,它需要额外的数据结构实现扫描顺序的转换.

4.2　基于游程的单次扫描CCA算法

一行连续连接的对象像素被称为游程,这些像素必定有相同标签. 基于游程技术处理图像,可以显著减少标签总数,有效降低解析标签等价关系的复杂度,加快算法处理速度. 基于游程的算法把游程视作一个超级像素,不再检测像素之间的连接关系,而是考虑游程之间的连接关系.

Trein等人[52]提出基于游程长度编码(Trein's Run Length Encoding,TRLE)的CCA算法,采用游程压缩图像,仅采用开始和结束位置描述每个游程,在一个时钟周期内只需要发送两个位置就可以代表游程中所有对象像素,加快了算法处理速度. 算法为每个游程分配一个临时标签,通过比较相邻行之间的游程关系确定标签等价关系. 为了正确解析一些复杂图像带来的合并标签等价关系,作者使用一个从旧标签指向新标签的指针,当试图更新旧标签的特征数据时,会因为指针指向新标签而最终把特征数据累加更新到新标签上.

Zhao等人[53]提出了基于游程单次扫描(Zhao's Run-based Single Pass,ZRSP)的CCA算法,在实时自动目标识别系统上实现了对位流图像数据的处理. 该算法以像素为扫描单元,以游程为标记单元,结合了像素和游程的优点,并设计多层索引结构保持相邻行之间标签的正确对应关系,但是在某些更复杂的图像中,仍然存在链式问题.

4.2.1　TRSP算法

Tang等人[54]提出的TRSP算法采用如图3(*d*)所示的扫描窗口标记游程,并基于链表(Linked list)结构解析标签等价关系,消除了链式效应. 为了降低内存需求,算法采用了简单且高效的标签回收策略,给游程分配的标签从0到$W/2$(W为图像宽度),当标签到达最大值($W/2$)时重新从0计数. 这种自动回收标签技术无需额外的数据结构跟踪标签,也无需额外的控制逻辑判断连通域是否标记完成,减少了算法的内存空间消耗并降低了算法计算复杂度. 算法实现过程如图7所示:首先为游程分配临时标签,再基于链表结构建立游程标签等价关系,然后根据标签等价关系计算对象特征数据,最后在标签失效前通过链表传递标签等价关系和特征数据. 图像填充过程为原始图像填充额外对象像素,是个可选过程.

TRSP算法引入Next,Head,Tail和Data四个表建立链表结构,其中Head和Next是主表,Tail和Data是次表,次表只能由主表Head索引. Data表存储连通域特征值数据,Head,Tail和Next表充当指针构建并处理标签等价关系. 其中Head和Tail指针将记录前一行的游程标签,在后续扫描中通过指针的转换实现游程标签转换,表示两行游程的标签等价. Next指针可以在游程

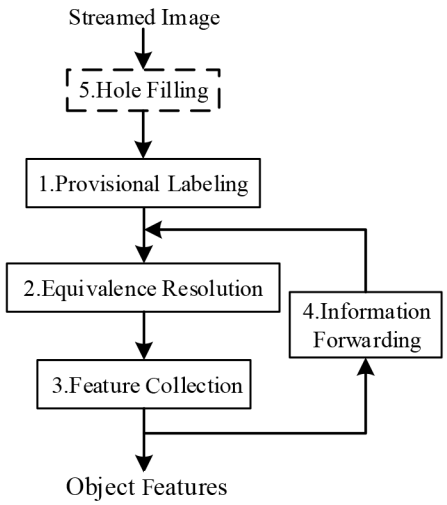


图7 TRSP算法概述

列表中插入新标签,将新标签附加到游程列表末尾以表示标签等价. 图8的示例说明指针的用法,图中前一行有标签L_p,当前行有等效游程列表$L_1 \rightarrow L_2 \rightarrow L_3$,假设它们属于同一个连通域,有以下定义:

(1)前一行标签的Head表示当前行等效游程列表的第一个元素,有$\mathrm{Head}[L_p]=L_1$;

(2)标签的Tail是等效游程列表的最后一个元素,有$\mathrm{Tail}[L_1]=L_3$,也等价于$\mathrm{Tail}[\mathrm{Head}[L_p]]=L_3$;

(3)Next指向游程列表中下一个元素,对于当前行等效游程列表,有$\mathrm{Next}[L_1]=L_2$,$\mathrm{Next}[L_2]=L_3$.

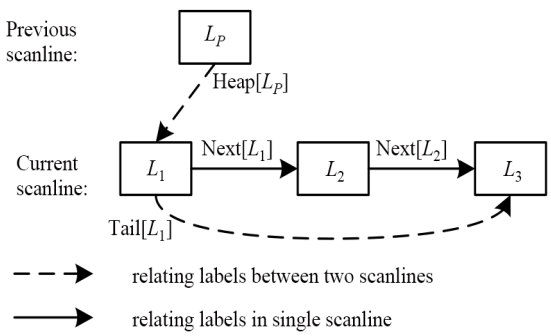


图8 Next, Head和Tail指针的用法

采用这种链表结构,只需对Head,Tail和Next表进行单次写入即可完成标签等价关系解析,完全避免了链式效应,实现了最大的吞吐量.

4.3 其它单次扫描CCA算法

本节主要分析两种算法:(1)Gu等人[56]提出的基于单元单次扫描(Cell-based Single Pass,CSP)的CCA算法;(2)Jeong等人[58]提出的单次扫描(Jeong's Single Pass,JSP)CCA算法.

4.3.1 CSP算法

CSP算法采用单元(Cell)压缩图像,与BCL[26]算法有相似之处,BCL算法采用2×2大小的块替换像素,CSP算法的单元大小为$n\times n$,并不局限于2×2. 对于每一个单元,只要单元中存在对象像素,那么整个单元就可以看成一个对象单元,当单元被分配标签后,单元中所有像素(无论是否为对象像素)都会获得相同标签.

当$N=nM$时,CSP算法把$N\times N$大小的像素图像压缩成$M\times M$大小的单元图像. CSP算法只考虑四连通性,假设$b(x,y)$表示坐标(x,y)的单元,其中$0\leqslant x\leqslant M-1$,$0\leqslant y\leqslant M-1$,当前单元$b(x,y)$对应标签表示为$L_X$,它上方单元$b(x,y-1)$对应标签表示为$L_{\mathrm{up}}$,它左边单元$b(x-1,y)$对应标签表示为$L_{\mathrm{left}}$,为当前单元选择标签如算法7所示.

算法7 CSP算法

IF $b(x,y-1)=b(x-1,y)=0$ THEN
  $L_X=L_{\mathrm{new}}$
ELSE IF $b(x,y-1)=1$, $b(x-1,y)=0$ THEN
  $L_X=L_{\mathrm{up}}$
ELSE IF $b(x,y-1)=0$, $b(x-1,y)=1$ THEN
  $L_X=L_{\mathrm{left}}$
ELSE IF $b(x,y-1)=b(x-1,y)=1$, $L_{\mathrm{up}}=L_{\mathrm{left}}$ THEN
  $L_X=L_{\mathrm{left}}$
ELSE $b(x,y-1)=b(x-1,y)=1$, $L_{\mathrm{up}}\neq L_{\mathrm{left}}$
  $L_X=\mathrm{Min}(L_{\mathrm{up}}, L_{\mathrm{left}})$
END IF

CSP算法采用重标记存储器(Relabeling Memory)记录和解析标签等价关系,在每行行尾根据记录结果从右到左将临时标签替换成新标签进行重新标记.

当$n=1$时,CSP算法与基于像素的CCA算法有相同结果,当$n>1$时,两者的结果将并不完全一致,因为CSP算法中被认为相互连接的对象单元中可能有并不相连的像素. 这是考虑算法计算复杂度和结果精确度之间的结果. 当$n>1$时,CSP算法的计算复杂度和内存消耗从$O(N^2)$降低到$O(M^2)$.

4.3.2 JSP算法

为了消除链式效应,JSP算法提出了一种无标签合并周期的硬件架构,如图9所示. 八连通检查器(8-Connectivity Checker)采用如图3(a)所示的扫描窗口,遵循OCL算法的标签选择策略为当前像素选择标签;信息提取器(Information Extractor)为标记连通域累加更新特征数据;标签堆栈(Label Stack)用来管理和回收标签,新分配标签从堆栈头部弹出,回收标签存放在堆栈尾部. 标签移位寄存器在合并发生后的下个时钟周期里把所有旧标签直接替换成新标签,因此无需解析标签等价关系.

该架构采用标签移位寄存器替换行缓冲区储存标签信息,相邻标签移位寄存器之间有一个条件选择器S_1,S_1的输入、输出在图9中用红色标出. 标签移位寄存

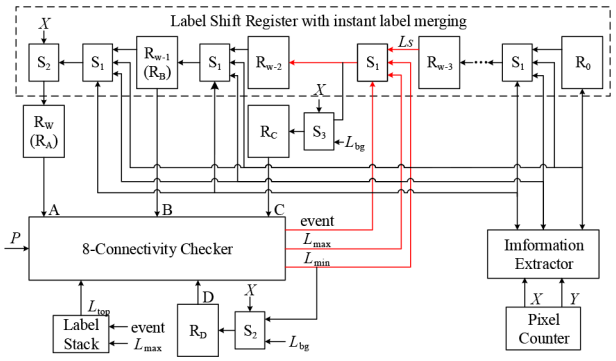


图9　JSP算法的硬件架构,其中红线表示S1的输入和输出

器中储存的标签在每个时钟周期向左移动,作为条件选择器输入标签L_S. 八连通检查器根据扫描窗口为当前像素分配标签并储存在标签移位寄存器R_0中,同时生成event信号和输入标签L_{max},L_{min}. 条件选择器在event信号控制下,动态地从三个输入标签中选择输出标签到下一个标签移位寄存器.

发生合并事件时,event信号输出1,合并中较大标签指定为L_{max},较小标签指定为L_{min},将它们输入到条件选择器中,对L_S与L_{max}进行比较,如果相同说明L_S是旧标签,输出L_{min},不相同则输出L_S. 对于非合并事件,event信号输出0,标签L_S在移位时不改变.

5　比较与分析

本节对上述几种CCA算法进行评估,这些算法都在它们提出的相应硬件架构中实现,在下文中提到算法简称即代表其硬件架构. 对于不同分辨率图像,CCA硬件架构的内存消耗大小是评价该架构实现位流图像处理的重要指标之一. 因此本文将从内存消耗和最坏情况吞吐量等方面分析这些算法,并比较它们的硬件架构性能. 这些算法数据都是根据它们报告的结果近似计算的,但不失一般性.

5.1　内存需求分析

表2比较了OSP[42],IOSP[45],SLSP[48],DLSP[50],ZZSP[51],TRSP[54]和JSP[58]算法对$W\times H$大小图像的内存需求;表3比较了OSP和JSP算法对不同分辨率图像的数据存储所需内存位数;图10比较了IOSP,SLSP,DLSP,ZZSP和TRSP算法对不同分辨率图像的内存总数.

表2中为了进行公平的比较,假设所有算法都是对$W\times H$像素大小图像进行处理,提取特征数据包括面积(Area)和边界框(Bounding box). 标签数量N_L是所有架构的关键因素,它直接决定了存储器的深度和宽度. 标签重用FIFO储存所有标签,它的大小为标签数量N_L乘以标签宽度W_L,标签宽度由$W_L=\log_2 N_L$计算得到. 行缓冲区RB(Row Buffer)缓存一行像素的临时标签,它的大

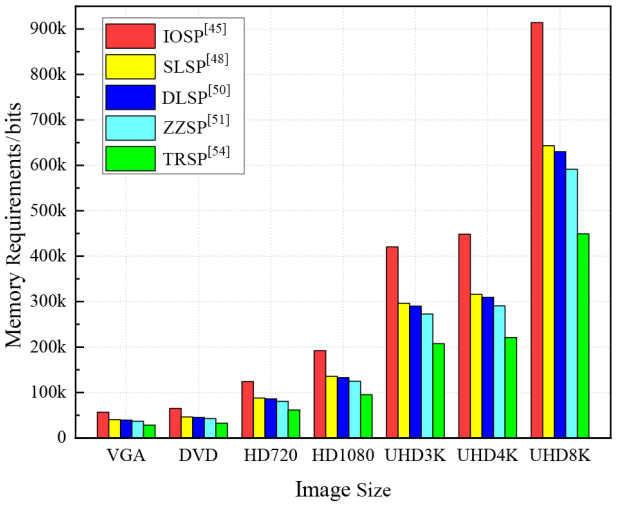


图10　几种CCA算法处理不同分辨率图像的内存位数比较

小为$W\times W_L$. 合并表MT记录标签之间的等价关系,它的深度为标签数量N_L,宽度为标签宽度W_L,大小为$N_L\times W_L$. 数据表DT储存标签对应连通域的特征数据,数据表深度为标签数量N_L,宽度为特征数据宽度W_D,大小为$N_L\times W_D$. 由于提取的特征数据包括面积和边界框,可以计算出特征数据宽度为$W_D=2\log_2 W+2\log_2 H+\log_2 WH$.

OSP算法不回收标签,对标签的计数只需简单的计数器而不采用FIFO寄存器,因此标签数量取决于图像大小,最大数目为$WH/4$. OSP算法在行尾进行反向解链,需要一个链堆栈S记录两个合并标签,因此链堆栈宽度为标签宽度的两倍,深度为一行图像中存在的最大标签合并次数$N_M=(W-1)/2$,链堆栈大小为$2\times W_L\times N_M$. 由于不考虑标签回收,OSP算法的标签数量比其他算法大的多,对应的合并表和数据表内存占用也就很大,它的内存位数量如表3所示.

JSP算法采用标签移位寄存器实现行缓冲区,通过条件选择器直接替换所有旧标签,无需解析标签等价关系. 根据文献[58]的报告,JSP算法虽然采用了标签回收技术,但只是回收合并中的较大标签,标签回收策略并不完整,使得标签数量可能与OSP算法一样大. 如表3所示,JSP算法的标签重用FIFO和OSP算法的合并表一样大,但OSP算法还需要一个链堆栈S记录合并标签,所以内存需求上OSP算法更大一些. 对比表3和图10可以发现,这两种算法的内存需求远远大于其他算法(800M对比900K),OSP算法内存需求是所有算法框架中最大的,JSP算法次之.

IOSP算法采用激进的重标记方法使标签数量减少到$W/2$,但这种重标记技术额外需要一个深度为N_L,宽度为W_L的转换表TT(Translation Table)管理上一行和当前行标签之间的关系,此外,还需要两个合并表和数据表分别记录两行标签和特征数据. 如表3和图10所示,IOSP算法与OSP算法相比显著减少了内存需求,但

表2 不同单次扫描CCA架构对于W×H大小图像的内存需求

缩写对应描述	缩写	OSP[42]	IOSP[45]	SLSP[48]	DLSP[50]	ZZSP[51]	TRSP[54]	JSP[58]
Number of label	N_L	$\frac{W\times H}{4}$	$\frac{W}{2}$	$\frac{W+5}{2}$	$\frac{W+5}{2}$	$\frac{W}{2}$	$\frac{W}{2}$	$\frac{W\times H}{4}$
Nunber of merge	N_M	$\frac{W-1}{2}$	$\frac{W-1}{2}$	$\frac{W-1}{2}$	$\frac{W-1}{2}$	—	—	—
Row buffer	RB	$W_L\times W$	$W_L\times W$	$W_L\times W$	$W_L\times W$	$W_L\times W$	$2\times W$	$W_L\times W$
Merge table	MT	$W_L\times N_L$	$2W_L\times N_L$	$W_{AL}\times N_L$	$W_{AL}\times N_L$	$W_{AL}\times N_L$	—	—
Data table	DT	$W_D\times N_L$	$2W_D\times N_L$	$W_D\times N_L$	$W_D\times N_L$	$W_D\times N_L$	$W_D\times N_L$	$W_D\times N_L$
Chain stack	S	$2W_L\times N_M$	$2W_L\times N_M$	$2W_L\times N_M$	$2W_L\times N_M$	—	—	—
Recycle FIFO	FIFO	—	—	$W_L\times N_L$	$W_L\times N_L$	$W_L\times N_L$	—	$W_L\times N_L$
Translation table	TT	—	$W_L\times N_L$	—	—	—	—	—
Stale label stack	SLS	—	—	$W_L\times\frac{W}{10}$	—	—	—	—
Valid tag	V	—	—	N_L	—	—	—	—
Active tag	AT	—	—	$2\times N_L$	$2\times N_L$	$\left(\log_2 W+1\right)\times N_L$	—	—
Linked list	List	—	—	—	—	—	$3W_L\times N_L$	—
Zig-Zag buffer	ZZ	—	—	—	—	W	—	—

注：标签宽度(The width of label)，$W_L=\log_2 N_L$；增强标签宽度(The width of augmented label)，$W_{AL}=W_L+\log_2 H$；宽度数据宽度(The width of the data)，$W_D=2\log_2 W+2\log_2 H+\log_2 WH$

表3 OSP和JSP算法架构对不同分辨率图像的内存需求比较

图像大小	VGA	DVD	HD720	HD1080	UHD3K	UHD4K	UHD8K
	640×480	720×576	1 280×720	1 920×1 080	3 840×2 160	4 096×2 160	7 680×4 320
N_L	76 800	103 680	230 400	518 400	2 073 600	2 211 840	8 294 400
N_M	319	359	639	959	1 919	2 047	3 839
W_L	17	17	18	19	22	22	23
W_D	57	59	62	65	72	72	77
OSP[48]							
S	10 846	12 206	23 004	36 442	84 436	90 068	176 594
RB	10 880	12 240	23 040	36 480	84 480	90 112	176 640
MT	1 305 600	1 762 560	4 147 200	9 849 600	45 619 200	48 660 480	190 771 200
DT	4 377 600	6 117 120	14 284 800	33 696 000	149 299 200	159 252 480	638 668 800
Total	5 704 926	7 904 126	18 478 044	43 618 522	195 087 316	208 093 140	829 793 234
JSP[58]							
RB	10 880	12 240	23 040	36 480	84 480	90 112	176 640
FIFO	1 305 600	1 762 560	4 147 200	9 849 600	45 619 200	48 660 480	190 771 200
DT	4 377 600	6 117 120	14 284 800	33 696 000	149 299 200	159 252 480	638 668 800
Total	5 694 080	7 891 920	18 455 040	43 582 080	195 002 880	208 003 072	829 616 640

在图10所示的算法中，IOSP算法内存需求仍是最大的.

SLSP算法通过回收标签使标签数量N_L依赖图像宽度W，额外的5个时钟周期延迟用于补偿标签回收处理操作. 增强标签使用行号扩展了标签宽度，变为$W_{AL}=W_L+\log_2 H$. 算法的标签回收策略采用标签重用FIFO寄存器管理标签，需要增加一个两位比特的活动标签AT跟踪连通域标记是否完成，还需要一个一位比特的有效性标记V跟踪标签是否为过期标签，过期标签储存放在过期标签堆栈SLS中. 算法中证明过期标签的最大数量为$W/10$[48]，即SLS深度也为$W/10$. 与IOSP算法相比，SLSP算法只需要一个合并表和数据表，由于标签宽度变宽，合并表内存大小几乎相同，但数据表内存缩小一半，总内存需求显著减少了约28%.

DLSP算法通过双重查找解决了过期标签问题，无需过期标签堆栈SLS和有效性标记V，进一步减少内存需求，但这两个辅助数据结构内存消耗都比较小，因此提升有限，内存需求只减少约2%.

ZZSP算法需要额外的储存器结构实现“Z”字形顺序扫描，在扫描期间动态解析标签等价关系，避免了链堆栈S的需要. 算法中采用更大的活动标签AT记录横

坐标x(最大为图像宽度W)和纵坐标y的最低有效位(一位比特). 与DLSP算法相比内存量减少了约6%.

TRSP算法采用了两个行缓冲区,但行缓冲区不储存临时标签,宽度仅为一比特. 算法中还采用了4个表,每个表的深度最大为N_L,由Head,Next和Tail表组成的链表结构替换合并表解析标签等价关系,表位宽为W_L,Data表位宽为W_D. 算法采用的标签回收技术避免了标签重用FIFO和活动标签AT的需要,有效减少内存需求,与SLSP算法相比减少了大约30%,与ZZSP算法比减少了大约25%.

5.2　硬件架构比较

表4比较了OSP[42],IOSP[45],SLSP[48],ZZSP[51],TRSP[54]以及JSP[58]硬件架构的实现结果,由于各种架构实现FPGA技术不同、处理图像分辨率不同和提取特征参数不同,为了实现公平比较,比较结果只包括CCA硬件架构,不考虑图像采集和外部接口模块. 为了分析不同架构的性能差异,采用如公式(5)所示的线性函数对原始数据进行归一化处理:

$$X_{\text{norm}}=\frac{X-X_{\min}}{X_{\max}-X_{\min}} \tag{5}$$

其中X为原始数据,$X_{\max}$和$X_{\min}$为原始数据中最大值和最小值.

由于LUT(逻辑查找表)和寄存器都是由FPGA内部逻辑查找表综合生成,所以衡量FPGA寄存器资源消耗可以将二者累加综合考虑. 归一化结果在LUT、寄存器和BRAM方面越小越好,最大工作时钟频率越大越好.

表4　几种CCA硬件架构的比较

硬件架构	采用设备	图像大小/pixel	特征提取[a]	LUTs	寄存器	BRAM /bit	Fmax/MHz	吞吐量/(Mpixel/s)
SLSP[48]	Kintex 7	256×256	BB	493	296	108K	185.59	154.50
ZZSP[51]	Kintex 7	256×256	BB+A	882	503	18K	220.02	220.02
TRSP[54]	Virtex 2	256×256	BB	547	183	72K	104.26	104.26
OSP[42]	Spartan 2	640×480	A,N	810	286	16K	—	—
IOSP[45]	Virtex 2	640×480	A,N	1 757	600	72K	40.64	38.25
TRSP[54]	Virtex 2	640×480	BB	654	227	92K	97.07	97.07
JSP[58]	Cyclone 4	640×480	BB+C	36 478	—	18K	60.58	60.58

注:a 特征向量的缩写:(A)面积;(BB)边界框;(C)质心;(N)连通域数量

在处理256×256分辨率图像时,SLSP架构消耗了最多BRAM,归一化结果为1.0,LUT和寄存器消耗为0.09,最大工作时钟频率为0.7. ZZSP架构采用LUT作为分布式RAM实现小型数据结构需要消耗更多LUT,但显著减少了BRAM,此外,它的最大工作时钟频率也是最高的,各项性能(LUT,BRAM和最大工作时钟频率)归一化结果分别为1.0,0.0和1.0. TRSP架构采用简化逻辑的链表结构构建标签等价关系,只需要很少的寄存器和LUT资源,但是最大工作时钟频率最低,它的各项性能归一化结果为0.0,0.6和0.0. 在这三种算法架构中,ZZSP架构不仅有最少的BRAM和最高的最大工作时钟频率,且在寄存器资源消耗上也不多,综合性能是最优的.

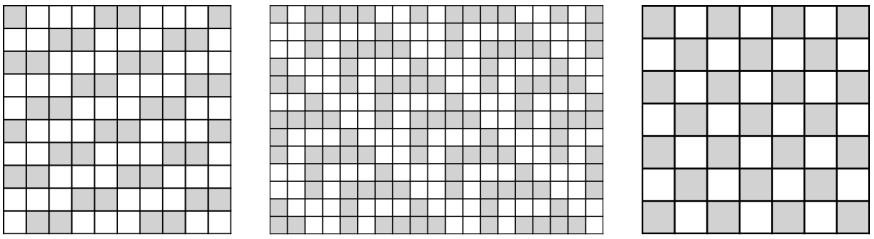
在处理640×480分辨率图像时,OSP架构消耗较少硬件资源是因为处理图像中仅包含255个标签,消耗LUT和BRAM的归一化结果为0.006和0.0. IOSP架构需要两个合并表和数据表记录标签等价信息和数据信息,使得LUT和寄存器单元消耗较OSP架构翻了一倍,它的各项性能(LUT,BRAM和最大工作时钟频率)归一化结果为0.04,0.737和0.0. JSP架构为了在合并期间替换所有旧标签,采用标签移位寄存器和多路选择器实现标签更新,多路选择器通过逻辑运算输出标签,导致架构消耗的LUT资源远远大于其他架构,并降低最大工作时钟频率,它的各项性能归一化结果为1.0,0.026和0.353. TRSP架构在四种架构中消耗最少的寄存器资源并有最高的最大工作时钟频率,各项性能归一化结果为0.0,1.0和1.0. 虽然TRSP架构也消耗了最多的BRAM,但是综合性能超过了其他三种架构.

综合上述分析,ZZSP架构的综合性能是所有架构中最优的.

5.3　最差情况吞吐量

架构的吞吐量分为两部分:一部分为每个时钟周期处理一个像素的静态部分,这使得算法的吞吐量直接受限于硬件架构工作时钟频率;另一部分为根据图像内容解析等价关系的数据相关部分,这将会影响架构处理单个像素的平均周期. 各架构在最差情况下处理像素的平均周期最大,决定了处理时间的上限,因此比较最差情况下架构的吞吐量有一定意义. 图11给出了CCA算法对应最差吞吐量图案. 如图11(a)所示的阶梯图案将会产生最大的链堆栈,OSP[42],SLSP[48]和DLSP[50]架构在行尾解链会额外增加$W/5$个时钟周期消耗,这些架构处理一个像素需要6/5个时钟周期.

IOSP[45]架构中许多合并是由转换表决定的,使得创建最大链堆栈的模式变得更加复杂,如图11(b)所示



(*a*) 阶梯图案 (*b*) 羽毛图案 (*c*) 棋盘图案

图11 造成最差吞吐量的图案

的羽毛图案中两行像素增加 $W/8$ 个时钟周期消耗，IOSP架构处理一个像素需要17/16个时钟周期.

如图11(*c*)所示的棋盘图案限制了游程的优势，是所有基于游程算法的最差情况. TRSP[54]架构虽然是基于游程算法，但它和ZZSP[51]、JSP[58]架构都不需要额外时钟周期处理标签等价关系，因此它们处理一个像素只需要一个时钟周期. 如表4所示，TRSP、JSP和ZZSP架构的吞吐量和工作时钟频率相同.

5.4 综合分析

优秀的处理算法不仅要实现实时处理，即需要以每个时钟周期一个像素的速度处理位流像素，还要合理使用硬件资源.

在基于像素的CCA算法中，OSP[42]，IOSP[45]，SLSP[48]和DLSP[50]算法解链延迟取决于图像连通域的几何复杂度，这导致算法吞吐量不一致. 对于一般图像，解链延迟通常小于1%[51]；但对于最差情况，单行延迟可达50%，整幅图像延迟为20%，使得算法无法实现实时处理. ZZSP[51]算法采用“Z”字形扫描避免解链延迟，实现了实时处理. 虽然采用“Z”字形扫描和多次查找合并表使得该算法需要比光栅式扫描算法更多的逻辑资源，但是总体资源消耗仍然较少. 因此ZZSP算法是最优的基于像素的单次扫描CCA算法.

TRSP[54]算法通过链表结构操纵指针管理标签合并实现实时处理且消耗硬件资源较少，是最优的基于游程的单次扫描CCA算法. 然而，TRSP算法是四连通域检测算法，在检测精度上落后于八连通的ZZSP算法. 但四连通扩展为八连通并不困难，因此TRSP算法仍然是较优的CCA算法.

JSP[58]算法也可以实现实时处理，但是与上述算法不同，它采用标签移位寄存器结构进行数据存储消耗了大量逻辑资源并导致时钟频率较低，此外，算法中回收标签不完整，导致内存需求很大，这使得JSP算法的性能与ZSSP和TRSP算法相比相差较大.

6 结论与展望

本文分析了近十年以来国内外发展的CCL算法和CCA算法，介绍了主要CCA算法的实现策略和架构，还介绍了适合CCA算法的等价解析策略，并比较了几种主要算法的内存需求、硬件架构和延迟. 在内存方面，OSP和JSP算法的内存需求远大于其他算法，TRSP算法的内存需求最小；在硬件架构方面，ZZSP算法实现了最大的工作时钟频率；在延迟方面，ZZSP，TRSP和JSP算法的每秒吞吐量和时钟频率相同，都可以实现实时处理. 综上所述，ZZSP和TRSP算法是最优的单次算法CCA算法. 本文分析结论为实现基于FPGA高速位流图像的连通域检测提供了理论依据和数据参考.

关于CCA问题的相关工作，未来可以从以下几个方面进一步展开：

(1)寻找更加有效的标签等价关系解析方法，当前联合查找算法中的展平操作中会增加延迟，如果能够在扫描过程中直接解析等价关系而不用等到行尾，那么就可以减少延迟，提高算法处理速度.

(2)在多块FPGA上并行实现CCA算法，进一步提高基于硬件架构的CCA算法处理速度.

(3)对现有CCA算法和硬件架构进一步改进和融合，使其能够更有效地提取图像中连通域特征.

(4)在当前CCA算法中，提取连通域的特征只包括面积、边界框和质心，设计新的CCA算法提取图像的周长、圆度和欧拉数等更多特征.

(5)设计适用于超大分辨率图像的处理算法，可以考虑将超大图像分割成多块，对分割的多块子图像进行标记处理.

(6)可以结合现代工业对三维图像检测的需要，将CCA算法的设计从二值图像处理扩展到三维图像处理. 通过研究分析三维图像像素之间的连通关系，实现对三维图像的CCA算法的高效并行化实现.

参考文献

[1] CHEN W J, GIGER M L, BICK U. A fuzzy c-means (FCM)-based approach for computerized segmentation of breast lesions in dynamic contrast-enhanced MR Images [J]. Academic Radiology, 2006, 13(1): 63-72.

[2] DING M, CAO Y F, WU Q X. Autonomous craters detec-

tion from planetary image[C]//Proceedings of the 3rd International Conference on Innovative Computing Information and Control (ICICIC 2008). Dalian: IEEE Computer Society, 2008: 443-443.

[3] ABUZAGHLEH O, BARKANA B D, FAEZIPOUR M. Noninvasive real-time automated skin lesion analysis system for melanoma early detection and prevention[J]. IEEE Journal of Translational Engineering in Health and Medicine, 2015, 3: 1-12.

[4] LAO W L, HAN J G, DE P. Automatic video-based human motion analyzer for consumer surveillance system[J]. IEEE Transactions on Consumer Electronics, 2009, 55(2): 591-598.

[5] JOSHI K A, THAKORE D G. A survey on moving object detection and tracking in video surveillance system[J]. International Journal of Soft Computing and Engineering, 2012, 2(3): 44-48.

[6] CHENG H Y, WENG C C, CHEN Y Y. Vehicle detection in aerial surveillance using dynamic bayesian networks[J]. IEEE Transactions on Image Processing, 2012, 21(4): 2152-2159.

[7] 曲立国,黄友锐,唐超礼,等. 基于FPGA的线阵CCD雨滴图像快速连续识别方法[J]. 光电工程, 2012, 39(10): 103-110.
QU L G, HUANG Y R, TANG C L, et al. Rapid and continuous identification method of linear array ccd dynamic raindrop image based on FPGA[J]. Opto-Electronic Engineering, 2012, 39(10): 103-110.

[8] MOHAMED A A, YAMPOLSKIY R V. An improved LBP algorithm for avatar face recognition[C]//Proceeding of the XXIII International Symposium on Information, Communication and Automation Technologies(ICAT). Sarajevo, Bosnia and Herzegovina: IEEE Computer Society 2011: 1-5.

[9] 华彩成, 王世凯, 张成峰, 等. 返回散射电离图仰角信息提取方法研究[J]. 电子学报, 2019, 47(11): 2438-2442.
HUA C C, WANG S K, ZHANG C F, et al. Research on Extraction Method of Elevation from Backscatter Lonogram[J]. Acta Electronica Sinica, 2019, 47(11): 2438-2442.

[10] ROSENFELD A, PFALTZ J L. Sequential operations in digital picture processing[J]. Journal of the ACM, 1966, 13: 471-494.

[11] LUMIA R, SHAPIRO L, ZUNIGA O. A new connected components algorithm for virtual memory computers[J]. Computer Vision Graphics and Image Processing, 1983, 22(2): 287-300.

[12] RONSE C, DEJVIJVER P. Connected Components In Binary Images: The Detection Problems[M]. Letchworth: Hertfordshire Research Studies Press, 1984.

[13] HARALICK R M. Some neighborhood operations[C]// Rosenfeld A. Real-Time Parallel Computing Image Analysis. New York: Plenum Press, 1981: 11-35.

[14] SUZUKI K, HORIBA I, SUGIE N. Linear-time connected-component labeling based on sequential local operations[J]. Computer Vision and Image Understanding, 2003, 89(1): 1-23.

[15] CHANG F, CHEN C J, LU C J. A linear-time component-labeling algorithm using contour tracing technique[J]. Computer Vision and Image Understanding, 2004, 93(2): 206-220.

[16] CHANG F, CHEN C J. A component-labeling algorithm using contour tracing technique[C]//Proceedings of the Seventh International Conference on Document Analysis and Recognition(ICDAR 2003). Edinburgh: IEEE Computer Society, 2003: 741-745.

[17] SUN D, LIU Y. A new contour tracing algorithm in eight-connected binary images[C]//Proceedings of the 2010 Third International Joint Conference on Computational Science and Optimization. Huangshan: IEEE Computer Society, 2010: 249-253.

[18] LIU Y, ZHU M. A new contour tracing automaton in binary image[C]//Proceedings of IEEE international conference in computer science and automation engineering. Shanghai: IEEE Computer Society, 2011: 577-581.

[19] STEFANO L D, BULGARELLI A. A simple and efficient connected components labeling algorithm[C]//Proceedings of the 10th International Conference on Image Analysis and Processing. Venice: IEEE Computer Society, 1999: 322-327.

[20] APPIAH K, HUNTER A, DICKINSON P, et al. A run-length based connected component algorithm for FPGA implementation[C]//Proceedings of the 2008 International Conference on Field-Programmable Technology(FPT). Taipei: IEEE Computer Society, 2008: 177-184.

[21] HE L, CHAO Y, SUZUKI K. A run-based two-scan labeling algorithm[J]. IEEE Transactions on Image Processing, 2008, 17(5): 749-756.

[22] HE L, ZHAO X, CHAO Y, et al. Configuration-transition-based connected-component labeling[J]. IEEE Transactions on Image Processing, 2014, 23(2): 943-951.

[23] ZHAO X, HE L, YAO B, et al. A new connected-component labeling algorithm[J]. IEICE Transactions on Infor-

mation and Systems, 2015, 98(11): 2013-2016.
[24] HE L, CHAO Y, SUZUKI K. An run-based one-and-a-half-scan connected-component labeling algorithm[J]. International Journal of Pattern Recognition and Artificial Intelligence, 2010, 24(4): 557-579.
[25] WU K, OTOO E, SUZUKI K. Optimizing two-pass connected-component labeling algorithms[J]. Pattern Analysis and Applications, 2009, 12(2): 117-135.
[26] GRANA C, BORGHESANI D, CUCCHIARA R. Optimized block-based connected components labeling with decision trees[J]. IEEE Transactions on Image Processing, 2010, 19(6): 1596-1609.
[27] HE L, CHAO Y, SUZUKI K, et al. Fast connected-component labeling[J]. Pattern Recognition, 2009, 42(9): 1977-1987.
[28] HE L, CHAO Y, SUZUKI J. An efficient first-scan method for label-equivalence-based labeling algorithms[J]. Pattern Recognition Letters, 2010, 31(1): 28-35.
[29] ZHAO F, ZHANG Z Y. Hardware acceleration based connected component labeling algorithm in real-time ATR system[C]//Proceedings of SPIE 8784, Fifth International Conference on Machine Vision(ICMV 2012): Algorithms, Pattern Recognition, and Basic Technologies. Wuhan: Society of Photo-Optical Instrumentation Engineers, 2013: 87841S.
[30] CABARET L, LACASSAGNE L, OUDNI L. A review of world's fastest connected component labeling algorithms: speed and energy estimation[C]//Proceedings of the International Conference on Design and Architectures for Signal and Image Processing(DASIP). Madrid: IEEE, 2014: 1-6.
[31] CABARET L, LACASSAGNE L. What is the world´s fastest connected component labeling algorithm[C]//Proceedings of the 2014 IEEE Workshop on Signal Processing Systems(SiPS). Belfast: IEEE, 2014: 1-6.
[32] CHANG W Y, CHIU C C, YANG J H. Block-based connected-component labeling algorithm using binary decision trees[J]. Sensors, 2015, 15(9): 23763-23787.
[33] ROUABEH H, ABDELMOULA C, et. al. A new efficient connected component labeling algorithm and its VHDL circuit[C]//Proceedings of the 2016 28th International Conference on Microelectronics(ICM). Giza: IEEE Circuits and Systems Society, 2016: 105-108.
[34] GUPTA S, PALSETIA D, PATWARY M M A, et al. A new parallel algorithm for two-pass connected component labeling[C]//Proceedings of the 2014 IEEE International Parallel and Distributed Processing Symposium Workshops(IPDPSW). Phoenix: IEEE, 2014: 1355-1362.
[35] HE L, REN X, GAO Q, et al. The connected-component labeling problem: A review of state-of-the-art algorithms [J]. Pattern Recognition, 2017, 70: 25-43.
[36] NIU L Q, CHEN X, PENG M, et al. Connected components labeling based on union-find operations applied to connected branches[J]. Journal of Intelligent and Fuzzy Systems, 2017, 32(5): 3739-3748
[37] HOPCROFT J E, ULLMAN J D. Set merging algorithms [J]. SIAM Journal on Computing, 1973, 2(4): 294-303.
[38] TARJAN R E, LEEUWEN J V. Worst-case analysis of set union algorithms[J]. Journal of the Assoaatton for Computmg Machinery, 1984, 31(2): 245-281.
[39] DILLENCOURT M B, SAMET H, TAMMINEN M. A general approach to connected-component labeling for arbitrary image representations[J]. Journal of the Assoaatton for Computmg Machinery, 1992, 39(2): 253-280.
[40] WU K, OTOO E, SHOSHANI A. Optimizing connected component labeling algorithms[C]//Proceedings of Image Processing. San Diego, California: The International Society for Optical Engineering, 2005: 1951-1976.
[41] ABUBAKER A, QAHWAJI R, IPSON S, et al. One scan connected component labeling technique[C]//Proceedings of the 2007 IEEE International Conference on Signal Processing and Communications(ICSPC 2007). Dubai: IEEE Computer Society, 2007: 1283-1286.
[42] BAILEY D G, JOHNSTON C T. Single pass connected components analysis[C]//Proceedings of Image and Vision Computing. New Zealand: IEEE, 2007: 282-287.
[43] Bailey D, Johnston C, Ma N. Connected components analysis of streamed images[C]//Proceedings of the 2008 International Conference on Field Programmable Logic and Applications(FPL). Heidelberg: IEEE Computer Society, 2008: 679-682.
[44] JOHNSTON C, BAILEY D. FPGA implementation of a single pass connected components algorithm[C]//Proceedings of the 4th International Symposium on Electronic Design, Test and Applications. Hong Kong: IEEE Computer Society, 2008: 228-231.
[45] MA N, BAILEY D G, JOHNSTON C T. Optimised single pass connected components analysis[C]//Proceedings of the 2008 International Conference on Field-Programmable Technology(FPT). Taipei: IEEE Computer Society, 2008: 185-192.
[46] KLAIBER M, ROCKSTROH L, WANG Z, et al. A mem-

ory-efficient parallel single pass architecture for connected component labeling of streamed images[C]//Proceedings of the 2012 International Conference on Field-Programmable Technology(FPT). Seoul: IEEE Computer Society, 2012: 159-165.

[47] KLAIBER M, BAILEY D, AHMED S, et al. A high-throughput FPGA architecture for parallel connected components analysis based on label reuse[C]//Proceedings of the 2013 International Conference on Field-Programmable Technology(FPT). Kyoto: IEEE Computer Society, 2013: 302-305.

[48] KLAIBER M, BAILEY D, BAROUD Y, et al. A resource-efficient hardware architecture for connected components analysis[J]. IEEE Transactions on Circuits and Systems for Video Technology, 2016, 26(7): 1334-1349.

[49] KLAIBER M, BAILEY D, SIMON S. A single-cycle parallel multi-slice connected components analysis hardware architecture[J]. Journal of Real-Time Image Processing, 2019, 16(4): 1151-1175.

[50] KLAIBER M, BAILEY D, SIMON S. Comparative study and proof of single-pass connected components algorithms[J]. Journal of Mathematical Imaging and Vision, 2019, 61: 1112-1134.

[51] BAILEY D, KLAIBER M. Zig-Zag based single-pass connected components analysis[J]. Journal of Imaging, 2019, 5(4): 45.

[52] TREIN, J, SCHWARZBACHER A T, HOPPE B, et al. Development of a FPGA based real-time blob analysis circuit[C]//Proceedings of the Irish Signals and Systems Conference. Derry: IEEE, 2007: 121-126.

[53] ZHAO F, LU H Z, ZHANG Z Y. Real-time single-pass connected components analysis algorithm[J]. EURASIP Journal on Image and Video Processing, 2013, 1: 21.

[54] TANG J W, SHAIKH H N, SHEIKH U U, et al. A linked list run-length-based single-pass connected component analysis for real-time embedded hardware[J]. Journal of Real-Time Image Processing, 2018, 15(1): 197-215.

[55] GU Q Y, TAKAKI T, ISHII I. 2000-fps multi-object extraction based on cell-based labeling[C]//Proceedings of the 2010 IEEE International Conference on Image Processing(ICIP 2010). Hong Kong: IEEE, 2010: 3761-3764.

[56] GU Q, TAKAKI T, ISHII I. A fast multi-object extraction algorithm based on cell-based connected components labeling[J]. IEICE Transactions on Information and Systems, 2012, E95.D(2): 636-645.

[57] GU Q, TAKAKI T, ISHII I. Fast FPGA-based multi-object feature extraction[J]. IEEE Transactions on Circuits and Systems for Video Technology, 2013, 23(1): 30-45.

[58] JEONG J, LEE G, LEE M, et al. A single-pass connected component labeler without label merging period[J]. Journal of Signal Processing Systems, 2016, 84(2): 211-223.

[59] ISHII I, TATEBE T, GU Q Y, et al. 2000 fps real-time vision system with high-frame-rate video recording[C]//Proceedings of the 2010 IEEE International Conference on Robotics and Automation(ICRA 2010). Anchorage: IEEE, 2010: 1536-1541.

[60] KUMAR V S, IRICK K, MAASHRI A A, et al. A scalable bandwidth aware architecture for connected component labeling[C]//Proceedings of the 2010 IEEE Computer Society Annual Symposium on VLSI(ISVLSI). Lixouri: IEEE, 2010: 116-121.

[61] 赵峙江,张田文,张志宏. 一种基于视觉模型与连通域统计的阈值分割新算法[J]. 电子学报, 2005, 33(5): 793-797.
ZHAO S J, ZHANG T W, ZHANG Z H. A Novel Threshold Algorithm about Image Segmentation Based on Vision Model and Statistics of Consecutive Fields[J]. Acta Electronica Sinica, 2005, 33(5): 793-797.

[62] MALIK A W, THÖRNBERG B, IMRAN M, et al. Hardware architecture for real-time computation of image component feature descriptors on a FPGA[J]. International Journal of Distributed Sensor Networks, 2014(1): 1-14.

[63] YU Z Q, CLAESEN L, PAN Y, et al. SoC processor for real-time object labeling in life camera streams with low line level latency[C]//Proceedings of the 2014 IEEE International Symposium on Circuits and Systems(ISCAS). Melbourne: IEEE, 2014: 345-348.

[64] TAYARA H, HAM W, CHONG K T. A real-time marker-based visual sensor based on a fpga and a soft core processor[J]. Sensors, 2016, 16(12): 2139.

[65] LING L, CHEN Z, LI S, et al. FPGA-based connected components analysis algorithm without equivalence-tables [C]//Proceedings of the 10th International Conference on Intelligent Robotics and Applications(ICIRA 2017). Wuhan: Lecture Notes in Computer Science, 2017: 543-553.

[66] SPAGNOLO F, FRUSTACI F, PERRI S, et al. An efficient connected component labeling architecture for embedded systems[J]. Journal of Low Power Electronics and Applications, 2018, 8(1): 7.

[67] SPAGNOLO F, PERRI S, FRUSTACI F, et al. Connected component analysis for traffic sign recognition embedded processing systems[C]//Proceedings of the 2018

IEEE 25th International Conference on Electronics, Circuits and Systems(ICECS), Bordeaux: IEEE, 2018: 749-752.

[68] SPAGNOLO F, PERRI S, CORSONELLO P. An efficient hardware-oriented single-pass approach for connected component analysis[J]. Sensors, 2019, 19(14): 3055.

作者简介

曲立国 男,1979年出生,吉林舒兰人. 安徽师范大学物理与电子信息学院副教授. 主要研究方向为智能信息处理,检测技术与自动化装置.

E-mail: qlg77@163.com

陈国豪 男,1997年出生,安徽池州人. 安徽师范大学物理与电子信息学院硕士研究生. 主要研究方向为计算机视觉与自动化装置.

E-mail: cgh591614156@163.com

胡 俊 男,1996年出生,安徽六安人. 安徽师范大学物理与电子信息学院硕士研究生. 主要研究方向为智能信息处理与自动化装置.

E-mail: 17856939565@163.com

陈 鹏 男,1975年出生,安徽萧县人. 安徽师范大学物理与电子信息学院讲师. 主要研究方向为机器学习.

E-mail: ahnuchp@ahnu.edu.cn