基于行程累加的目标标记硬件加速算法

郑 红 李 俊

(北京航空航天大学 自动化科学与电气工程学院, 北京 100191)

摘 要: 针对目标标记算法的长耗时、大延迟严重影响目标跟踪的效果和实时性问题,提出了快速的基于行程累加的目标标记硬件加速算法. 通过对像素进行行程编码,按照定义的累加规则对相邻行的行程进行累加运算,避免了标号冲突的发生,无需建立等价表,实现了一遍扫描完成连通域标记和连通域特征的计算. 实验结果表明提出的算法资源占用较少,输出延迟小,与软件算法相比具有很高的加速比,有利于后续目标跟踪的高速实时处理.

关 键 词: 并行计算; 目标标记; 硬件加速; 行程累加

中图分类号: TP 391.42

文献标识码: A **文 章 编 号**: 1001-5965(2012)11-1425-05

Accelerated hardware target labeling based on run-length accumulation

Zheng Hong Li Jun

(School of Automation Science and Electrical Engineering, Beijing University of Aeronautics and Astronautics, Beijing 100191, China)

Abstract: To solve the problem that long time consuming and delay of target labeling algorithm seriously affected the effect and real-time performance of target tracking, fast accelerated hardware target labeling algorithm based on run-length accumulation was proposed. The pixels of image were converted to runs, and then the two runs of adjacent lines were accomplished accumulation calculation according to the defined accumulation rulers. The labeling conflicts were avoided. The algorithm could complete connected components labeling and calculate connected components features in a single pass without establishing equivalence table. Experiments show that the algorithm uses less memory resources with smaller result delay, and has higher speed ratio compared with software algorithms. So it can benefit to the fast and real-time processing of later target tracking.

Key words: parallel computation; target labeling; accelerated hardware; run-length accumulation

目标标记是从一幅包含运动目标的二值化图像中提取这些运动目标的位置大小等信息,它是目标识别与跟踪的一个重要的步骤. 标记算法性能的好坏直接影响跟踪的效果和实时性. 因此,研究高效的目标标记算法,以较小代价实现目标的实时跟踪具有重要的意义.

常用连通域标记算法来进行目标提取操作. 连通域标记方法又分为基于像素的标记和基于行程编码的标记方法. 基于像素标记的连通域算法比较成熟,它对图像进行初始标记,再结合记录标记冲突的等价表进行二次扫描得到标记结果[1-3]. 算法需要对图像进行缓存,并且需要较大的存储资源缓存等价表,带来了较大的资源消耗和较长的时间延迟.

为了加速算法的运行,提出了基于 FPGA (Field Programmable Gate Arrays)的连通域标记算法[4-6],利用硬件的并行执行在不同程度提高了标记算法的运算速度. 围绕减少标记算法运行时间、存储空间和降低标记结果输出延迟方面的研究一直没有停止[7-10],因为大多数算法需花费较

收稿日期: 2011-07-01; **网络出版时间**: 2012-11-16 11:07

网络出版地址: www.cnki.net/kcms/detail/11.2625.V.20121116.1107.006.html

基金项目: 光电信息重点实验室基金资助项目(9140C150105100C1502); 国家自然科学基金资助项目(60543006)

作者简介: 郑 红(1961-),女,河北邯郸人,教授, julyanna@vip.sina.com.

多的时间和存储空间,使其实际应用受到限制. 对目标像素的行程进行标号是加速连通域标号的常用方法[11].

行程编码的主要优点是存在计算图像区域的交、并的简单算法,并且行程编码算法产生的等价表很小,节约了大量的存储空间. 相对于基于像素扫描的标记算法,该算法运行速度更快. 行程编码的并行性适合在硬件上实现,文献[12]提出的行程编码的硬件加速算法在行人检测中也取得了较好的效果,但算法在解决复杂连通域标记时,需要花费很大的资源和延迟代价. 同时,算法还是需要存储等价表以进行二次扫描,使算法性能受到了一定影响.

为此,本文提出了基于行程累加的目标标记硬件加速算法. 该算法对相邻行的行程进行累加运算,无需建立等价表,也不需要进行二次扫描就能得到完整的目标标记信息. 能方便得到目标的各种特征信息,以较少资源代价实现了目标连通域的实时标记.

1 行程累加的目标标记算法

1.1 像素行程编码及行程行映射表

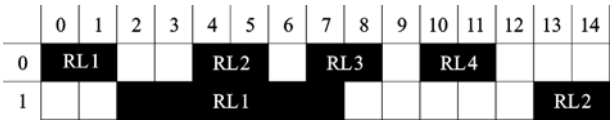
对图像的某行像素进行行程编码时,需要一个数据结构来表示一个行程. 这里采用行程的标号 ID、行程的起始列 ST 和终止列 EN、行程的起始行 RS 和终止行 RE 以及行程的连通标志 FG 作为行程的特征. 连通标志 FG 为 1 表示当前行程与上一行某个行程是连通关系,0 表示没有连通. 在进行行像素行程编码时 FG 初始值设为 0. 像素行程编码的伪代码如下:

```
i←0   Run←0
if   I(x,y)=1 and Run=0 then
   ST_i←y
   Run←1
End if
if Run=1 and (I(x,y)=0 or x=N) then
   EN_i←y-1
   r_i←y
   RS←RE←x
   i←i+1
   Run←0
   Tab(x,y)←i
   End if
```

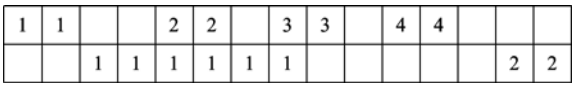
其中,I(x,y)表示图像的行像素,为二值图像;(x,y)表示图像的行列坐标;Run 为行程的标志;TAB(x,y)为行程行映射表,它是在行坐标的对应位置存储行程的标号 ID,对于背景像素,表中对应位置为 0.

对像素进行行程编码时,行程的起始行和终止行号相同. 行程编码以行为单位,每一行的行程 ID 都从 1 开始. 若当前像素为目标像素且行程标志为 0,则表示一个行程的开始. 若当前像素为一行的最后一个像素或者当前像素为背景像素同时行程标志为 1,则表示一个行程的结束. 在编码的同时,将行程的标号 ID 写入行程行映射表,将行程的特征写入行程缓冲 MEM(i). 图 1a 为行程示意图,图 1b 为行程编码时写入行程行映射表结果.

表 1 为图 1a 行程编码时写入行程缓冲中的各寄存器结果. 行程行映射表和行程缓冲有利于行程累加运算.



a 像素行程



b 行程映射表

图 1 行程编码及行程映射表示意图

表 1 行程缓冲中行程特征

序号	ID	ST	EN	RS	RE	FG
RL1	1	0	1	0	0	0
RL2	2	4	5	0	0	0
RL3	3	7	8	0	0	0
RL4	4	10	11	0	0	0
RL1	1	2	7	1	1	0
RL2	2	13	14	1	1	0

1.2 行程累加

定义行程累加运算如下:

1) 行程累加发生在相邻行独立的行程之间,当前行的行程缓冲根据上一行的行程行映射表决定何种运算,结果写入当前行行程缓冲.

2) 当前行的行程与上一行的行程之间关系有子集、并集和空集,分别如图 1a 中第 1 行的 RL1 与第 0 行的 RL2、第 1 行的 RL1 与第 0 行的 RL1、第 1 行的 RL2. 运算规则为

① 若当前行行程的起始列 STD 和终止列 END 之间对应上一行相同位置处有行程,则称此行程与上一行行程关系为子集,将上一行的行程标号 ID 传递到当前行行程,且当前行的 RSD 传递到上一行的 REU,并使 FG=1;

② 若当前行行程的起始列 STD-1 或终止列 END+1 对应上一行相同位置处有行程,则称此行程与上一行行程关系为并集. 从左至右,将上一行的行程标号 ID 传递到当前行行程,上一行行程的 ST 的最小值传递到当前行的 ST,上一行行程的 EN 最大值传递到当前行的 EN,当前行的 RSD 传递到上一行的 REU,并使 FG=1;

③ 若当前行行程起始列 STD 和终止列 END 之间对应上一行相同位置处没有行程,则两个行程之间关系为空集,当前行的行程数据结构不发生变化.

3) 若上一行某个行程没有参与累加运算,则此行程表示一个新的连通域.

每次累加运算后只保留当前行的行程缓冲和行程行映射表. 当下一行像素行程编码完成后,开始新的行程累加运算. 根据上述定义的行程累加规则,行程累加伪码如下:

```
i,j,m←0
if MEMD(m)≠0   then
  for t from MEMD(m)_ST-1 to MEMD(m)_
EN+1
  if TABU(t)≠0 then
  if MEMD(m)_ST<MEMU(TABU(t))_ST
    MEMD(m)_ST←MEMU(TABU(t))_ST
  else MEMD(m)_ST←MEMD(m)_ST
  if MEMD(m)_EN<MEMU(TABU(t))_EN
    MEMD(m)_EN←MEMU(TABU(t))_
EN
  else MEMD(m)_EN←MEMD(m)_EN
  MEMU(TABU(t))_FG←1
  MEMD(m)_FG←1
  MEMD(m)_RS←MEMU(TABU(t))_RS
  End if
 End for
End if
for i from 0 to totalrun
  if MEMU(i)_FG=0 then
    Connect(j)_left←MEMU(i)_ST
    Connect(j)_right←MEMU(i)_EN
    Connect(j)_top←MEMU(i)_RS
    Connect(j)_down←MEMU(i)_RE
    j←j+1
   End if
  End for
```

其中,MEMU 表示上一行的行程缓冲;MEND 表示当前行的行程缓冲;TABU 表示上一行的行程行映射表;TABD 表示当前行的行程行映射表;Connect_top,Connect_down,Connect_left,Connect_right 表示一个连通域的上、下、左、右边界信息;totalrun 为一行的行程总数. 表 2 所示为对图 1a 的行程累加运算结果.

图 2 所示为一幅大小为 $M\times N$ 的图像,行程个数为 $M\times N/4$. 对这种复杂连通域进行标记时,算法的执行时间和资源消耗都会大大增加,因此它是检验标记算法优劣的一个常用标准.

表 2　行程累加结果

序号	ID	ST	EN	RS	RE	FG
RL1	1	0	8	0	1	1
RL2	2	0	8	0	1	1
RL3	3	0	8	0	1	1
RL4	4	10	11	0	0	0
RL1	1	0	8	0	1	1
RL2	2	13	14	1	1	0

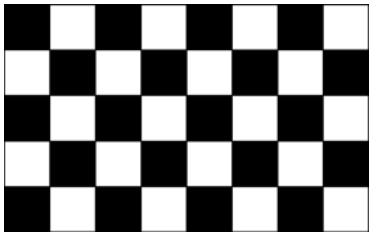


图 2　复杂连通域图

2　算法的 FPGA 实现

基于行程累加的连通域标记算法适合在硬件上流水实现. 图 3 为算法的硬件实现结构. 视频流按照光栅扫描的方式从上到下,从左到右进入行程编码模块,输出为目标的个数及目标的边界信息. 算法由像素行程编码、行程缓冲、行程行映射表、行程累加和标记结果输出 5 个模块组成.

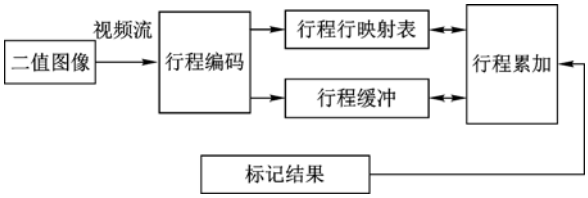


图 3　算法的硬件实现结构

2.1　像素行程编码

视频流按照以行为单位进行行程编码. 若当前像素是一行的第 1 个目标像素或者上一个行程之后的第 1 个目标像素,行程编码开始,此像素是新的行程的第 1 个像素,将此像素列坐标作为行程缓冲的 ST,行坐标作为行程缓冲的 RS,RE. 若像素是背景像素或者一行的最后一个像素,行程编码结束,此像素是行程的最后一个像素,将此像素列坐标作为行程缓冲的 EN.

2.2 行程缓冲

行程缓冲模块缓存有 2 个行缓存组成,分别缓存当前行和上一行的行程特征信息. 每个行程缓存由深度为 $N/2$、宽度为 41 位的双口 RAM 组成, N 为一行的像素个数,$N/2$ 为一行最大的行程个数. 一个行程的特征有 6 个元素 ID,ST,EN,RS,RW,FG 组成,FG 的位宽为 1 bit,其余每个元素的位宽为 8 bit.

2.3 行程行映射表

行程行映射表的长度为图像的宽度 N,宽度为 8 bit. 存储的内容为行程标号 ID,如图 1b 所示. 行程行映射表的作用是在进行行程累加时,直接由行程行映射表找出与当前行的某个行程进行运算的行程,不用搜索上一行的所有的行程,大大减小了搜索时间,加速了行程累加的计算.

2.4 行程累加模块

当前行的行程编码完成后,行程编码模块读取当前行的行程缓冲,顺序得到当前行的所有行程的特征,结合上一行的行程行映射表,按照行程累加规则进行运算,运算结果重新更新当前行的行程缓冲. 这个模块的触发是在第 2 行行程编码完成后进行的. 基于行程累加的标记算法,由行程的特征容易得到目标连通区域的边界信息.

2.5 标记结果输出

连通区域的边界信息由行程缓冲中的行程特征得到,进行累加后若某个行程的特征元素 FG 为 0,则此行程特征元素 ST,EN,RS,RE 即为连通区域的左、右、上、下边界信息. 采用宽度为40 bit,深度为 n 的 RAM 来存储标记结果,n 为设定的最大目标个数.

3 实验结果与分析

本文提出的基于行程累加的标记算法在硬件上进行了验证. 实验对图 2 所示复杂连通域和红外热像仪拍摄的天空云层背景飞机目标图像进行了标记. 标记复杂连通域是为了测试算法的鲁棒性,标记视频流图像是为了测试算法的实时性. 红外图像经过背景抑制及形态学滤波后得到包含目标的二值图像,对二值图像进行标记得到标记结果. FPGA 芯片采用 xilinx 公司的 XC5VFX70T,该芯片有 296 个 18 kB Block RAM 和 128 个硬核乘加器. 用来对比的算法是文献[1]所述经典的基于像素连通域方法及文献[12]提出的行程编码连通域标记,以下是对算法在资源需求、运行时间及延迟等方面的结果分析,并给出了标记结果.

3.1 资源需求

表 3 为标记图 2 时的硬件资源需求. 在标记红外实验图像时,算法需要的存储器资源与表 3 相同. 行程缓冲模块由 2 个行缓存组成,每个行缓存需要 40×160 bit 内存,总共需要 12 800 bit 的内存资源. 行程行映射表需要 320×8×2 bit 内存. 这两部分需要 1 个 Block RAM. 标记结果输出模块中 n 的个数取为 128,能满足实际需求,n 取得过大会造成不必要的资源浪费,需要 5 120 bit 的 RAM.

表 3 为标记图 2 时的硬件资源需求

名称	结果
FPGA 型号	xc5vfx70t
Slice LUTs 资源	1 079(2%)
Slice Registers 资源	542(1%)
BlockRAM/FIFO 资源	4 (3%)
Total Memory used/kB	144 (3%)

在 FPGA 中存储器资源是宝贵的,存储器资源是很多算法进行硬件加速时的遇到的主要瓶颈之一. 表 4 为 3 种算法在标记两种实验图像时的存储器资源对比. 由于基于行程累加的算法不需要对输入图像进行缓冲,也不需要对标记冲突图像建立等价表,节省了大量的存储器资源.

表 4 存储器资源对比 kB

序号	经典方法	行程编码	行程累加
图 2	6 375	862	144
图 4	3 648	684	144

3.2 运行时间及输出延迟

表 5 给出了标记图 4 红外图像及图 2 时的运行时间及延迟对比. 相比文献介绍的缓冲整幅图像然后进行标记方法,本文算法的运行时间和延迟要小很多. 第 1 行视频流输入时,开始执行像素行程编码,当前两行行程编码结束后,触发行程累加模块运行. 若某个行程累加结束,一个新的连通域就标记完成. 整幅图像输入后经过一行像素时钟周期的延迟就能得到全部连通域信息. 对图 2 进行标记时,文献所述方法的运行时间和延迟要比标记一般连通域大很多,而本文方法在标记图 2 图像时体现了很好的鲁棒性.

表 5 运行时间及延迟对比 ms

序号	经典方法	行程编码	行程累加
图 2 运行时间	35	22	12
图 4 运行时间	18	14	10
图 2 延迟时间	112	80	0.03
图 4 延迟时间	24	16	0.02

3.3 标记结果

将本文算法运用于实时标记天空背景飞机目标,图 4 所示是利用本算法的标记结果.

对于高帧频的红外视频流,算法能实时对运动目标进行标记,由标记结果很容易得出目标的边界信息(图中边框). 行程累加标记算法的资源消耗占系统资源比重很小,为特征提取,目标跟踪等后续算法留有充足的资源,算法的延迟较小为实时跟踪目标提供了有利条件.

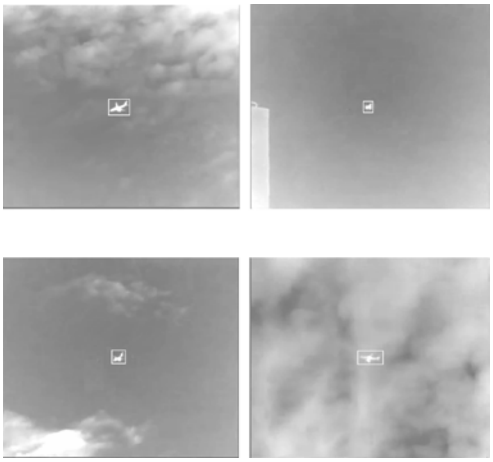


图 4　标记结果

4　结　论

本文提出的基于行程累加的连通域标记算法适合在硬件上以流水线结构并行实现,能流水输出标记结果,适合处理视频流图像标记. 通过引入行程行映射表和行程缓冲,缩短了算法的运行时间,使得算法的标记结果与原始图像数据之间的延迟很小. 行程的累加结果包含了目标的边界信息,且对行程的特征稍作修改,容易得出目标的其他灰度及形状信息. 在标记复杂连通域时,资源消耗和延迟并没有显著增加,具有良好的鲁棒性. 算法能实现高帧频图像的实时标记,对高速目标实时跟踪有一定的实际意义.

参考文献(References)

[1] Khanna V, Gupta P, Hwang C J. Finding connected components in digital images[C]//Henry Selvaraj. International Conference on Information Technology. Las Vegas: IEEE, 2005: 652-656

[2] 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

[3] Suzuki K, Horiba I, Sugie N. Fast connected component labeling based on sequential local operations in the course of forward raster scan followed by backward raster scan[C]//Bob Werner. 15th International Conference on Pattern Recognition. Barcelona: IEEE, 2000: 434-437

[4] Ito Y, Nakano K. Optimized component labeling algorithm for using in medium sized FPGAs[C]//John Hine. Ninth International Conference on Parallel and Distributed Computing. Dunedin: IEEE, 2008: 171-176

[5] 周平,王肖玉,刘越,等. 并行流水线实现的多红外标志点实时标记与识别新方法[J]. 仪器仪表学报, 2009, 30(6): 1240-1244
Zhou Ping, Wang Xiaoyu, Liu Yue, et al. New real-time labeling and recognition method for multiple infrared markers based on parallel pipeline[J]. Chinese Journal of Scientific instrument, 2009, 30(6): 1240-1244(in Chinese)

[6] Ma Ni, Bailey D G, Johnston C T. Optimised single pass connected components analysis[C]//Tarek Ghazawi. International Conference on Field-Programmable Technology. Taipei: IEEE, 2008: 185-192

[7] He Lifeng, Chao Yuyan, Kenji Suzuki. A linear-time two-scan labeling algorithm[C]//Jeff Rodriguez. International Conference on Image Processing. San Antonio: IEEE, 2007: 241-244

[8] He Lifeng, Chao Yuyan, Kenji Suzuki, et al. Fast connected-components labeling[J]. Pattern Recognition, 2009, 42(9): 1977-1987

[9] Roy Pradipta, Biswas P K. VLSI implementation of fast connected component labeling using finite state machine based cell network[C]//Jayanta Mukherjee. Computer Vision Graphics image Processing. Bhubaneswar: IEEE, 2008: 238-243

[10] Johnston Christopher T, Bailey Donald G. FPGA implementation of a single pass connected components algorithm[C]//Amine Bermark. IEEE International Symposium on Electronic Design. DELTA: IEEE, 2008: 228-231

[11] 孔斌. 快速连通域分析算法及其实现[J]. 模式识别与人工智能, 2003, 16(1): 110-115
Kong Bin. A fast connected component analysis algorithm and its implementation[J]. PR&AI, 2003, 16(1): 110-115(in Chinese)

[12] Appiah K, Hunter A, Dickinson P, et al. Accelerated hardware video object segmentation: From foreground detection to connected components labeling[J]. Computer Vision and Image Understanding, 2010, 114(11): 1282-1291

(编　辑:刘登敏)