文章编号:1671-4598(2018)02-0199-04 DOI:10.16526/j.cnki.11—4762/tp.2018.02.049 中图分类号:TP301.6 文献标识码:A

基于 Spark 的分布式车流量检测方法设计与实现

张 洪[1],赵 平[2],伍 玲[2],牛新征[3]
(1. 电子科技大学 信息与软件工程学院,成都 610054;2. 自贡市公安局交通警察支队,四川 自贡 643000;
3. 电子科技大学 计算机科学与工程学院,成都 611731)

摘要:针对现有车流量检测算法处理批量视频效率较低的问题,提出了基于 Spark 分布式计算框架的车流量检测方法;为提高计算节点对待处理数据的读取速率,在底层采用 HDFS 将交通视频元数据进行高效存储,同时设计了分布式帧差法以实现车流量并行检测与统计;最终,检测结果由主控节点存储于 HBase 中,以提升数据访问的可靠度;在真实数据集上的测试结果表明,与传统帧差法实现车流量检测算法相比较,处理速度提升 528%,同时准确率为 90.5%。

关键词:分布式计算框架;帧差法;车流量检测;计算机视觉

Design and Implementation of Distributed Traffic Flow Detection Method Based on Spark

Zhang Hong[1], Zhao Ping[2], Wu Ling[2], Niu Xinzheng[3]
(1. School of Information and Software Engineering, University of Electronic Science and Technology, Chengdu 610054, China; 2. Zigong City Public Security Bureau Traffic Police Detachment, Zigong 643000, China
3. School of Computer Science and Engineering, University of Electronic Science and Technology, Chengdu 611731, China)

Abstract: Aiming at the problem that the existing traffic flow detection algorithm is dealing with the low efficiency of batch video, a traffic flow detection method based on Spark distributed computing framework is proposed. In order to improve the reading rate of the data to be processed by the computing node, HDFS is used to store the traffic video metadata efficiently, and the distributed frame difference method is designed to realize the traffic detection and statistics. Finally, the test results are stored in the HBase by the master node to improve the access reliability of the data. The results of the test on the real data set show that compared with the traditional frame difference method to achieve traffic flow detection algorithm, the speed increase 528%, while the accuracy rate of 90.5%.

Keywords: distributed computing framework; frame difference method; traffic flow detection; computer vision

0 引言

近年来,社会经济发展的同时,推动了城市化道路的发展进程。城市道路中的车流量快速增长,促使智能交通系统(ITS, Intelligent Transport System)备受重视,基于视频的车流量检测[1-2]作为 ITS 的重要组成部分,它可以为城市道路的智能化调控(道路建设、红绿灯智能控制等)提供基础数据依据。将视频处理技术应用到车流量检测的方法主要有:帧差法[3]、背景差法[4]、光流法[5-6]等。

文献[5]检测精度高,但算法存在复杂度高以及计算量大等缺点。文献[3]改进传统帧差法实现检测车流量,在预设的检测带内检测出运动目标,并将运动目标转换成数据对象,根据数据对象的变化进行车辆检测。文献[6]采用改进的三帧差法与光流法结合实现检测运动对象的目的,从而减小处理的计算复杂度。文献[4]采用改进的背景差法实现运动目标检测,该算法对背景建模和阈值选取提出优化与改进,提升了检测准确率和应对复杂场景的能力。

交通视频数量的不断增加,无疑为车流量检测提出了更大的挑战。处理批量大视频数据的关键问题在于加快检测速度,随着云计算技术的不断发展,解决处理大视频数据的速度问题成为了可能,文献[7-8]将基于云的视频分析与 GPU 结合运用到交通监控系统中,成功的改善了视频监控的处理速度问题。同时,MPI、MapReduce[9]等并行计算模型以及 Hadoop、Spark[10]等大数据分布式计算框架的出现,为庞大且复杂的视频数据在普通 PC 机上完成计算提供了较好的支持,也为视频大数据提供良好的解决方案。文献[11]将 Spark 运用到大规模视频处理中,能较好的解决视频处理速度问题。

本文创新性的将 Spark 分布式计算框架与传统运动目标检测算法(帧差法)结合,设计并实现一种准确率达 90.5%,速度比传统帧差法快 528%的检测方法。在保证准确率的同时,改善传统帧差法在检测批量交通视频出现的处理速率慢的问题。

1 分布式车流量检测系统结构与原理

1.1 分布式车流量检测系统结构

分布式车流量检测系统分为数据存储层,数据计算层和系统应用层。数据存储层为数据计算层提供数据支持,数据计算层为系统应用层提供核心算法支持,系统应用层为系统提供应用服务支持。具体来说,数据存储作为数据输入层,提供车流

收稿日期:2017-07-05; **修回日期**:2017-08-07。
基金项目:四川省公安厅科研项目(2015SCYYCX06);成都市科学技术局软科学研究项目(2015—RK00—00247—ZF)。
作者简介:张 洪(1993-),男,四川南充人,硕士研究生,主要从事分布式云计算方向的研究。

量检测的数据输入。系统数据使用 HDFS 实现数据的分布式存储，以达到数据高可利用率和高容错性目的；数据计算作为数据逻辑处理层，提供车流量检测的核心算法实现。数据计算层使用帧差法完成车流量检测、目标跟踪与车辆统计功能，整个计算过程基于 Spark 分布式计算框架，将分布式车流量检测任务分配与计算节点选择都交给 Spark，以达到简化系统编程实现，以及从系统结构层面提升车流量检测速度的目的；系统应用作为视频处理结果的基本运用，提供车流量检测的核心应用，包括车流量检测和车辆信息持久化。车辆信息持久化基于 HBase 数据库，以达到提升数据的高可用性与高可靠性的目的。系统结构如图 1 所示。

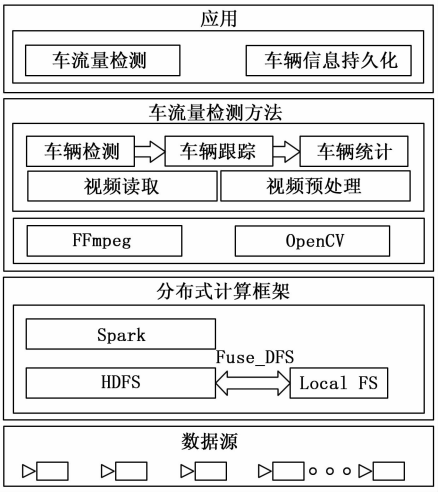

图 1 分布式车流量检测方法系统结构图

1.2 运动目标检测原理

视频目标检测与跟踪通常包括：视频图像采集、图像预处理、运动目标检测、运动目标跟踪、运动目标提取等主要部分，如图 2 所示。

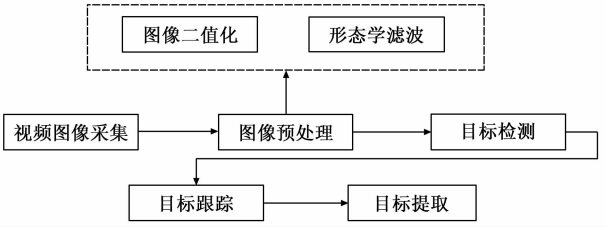

图 2 视频目标检测与跟踪

1.3 分布式车流量检测算法原理

本文设计的分布式车流量检测方法将 Hadoop 生态系统中的 HDFS 和 HBase 分别作为视频文件和处理结果的存储平台。考虑到传统视频目标检测算法的复杂度，本文选择帧差法用于运动目标检测，并采用运动目标位移量作为车辆跟踪的依据。按照算法的处理流程，方法主要分为 4 个模块：分布式视频获取模块，视频预处理与车辆检测模块，车辆跟踪模块，车流量统计入库模块。

2 分布式车流量检测方法设计与实现

本文提出的方法可分为 4 个部分：视频获取、图像预处理与车辆检测、车辆跟踪与统计、车辆信息持久化。车流量检测算法流程如图 3 所示，分布式任务执行流程如图 4 所示。具体实现步骤如下：

1）任务初始化：主控节点初始化车流量检测任务，并序列化任务发送给可用的计算节点；

2）视频获取：计算节点反序列化任务，获取待检测交通视频存储位置，并读取视频；

3）设置虚拟检测线：计算节点根据待处理视频大小设置虚拟检测线位置；

4）图像预处理与车辆检测：计算节点对待处理的两帧图像进行预处理，并完成车辆检测；

5）车辆跟踪与统计：计算节点根据当前帧的车辆位移实现车辆跟踪，并判断车辆是否经过虚拟检测线；

6）车辆信息持久化：计算节点将视频的检测结果返回主控节点，主控节点接收结果并持久化到 HBase。

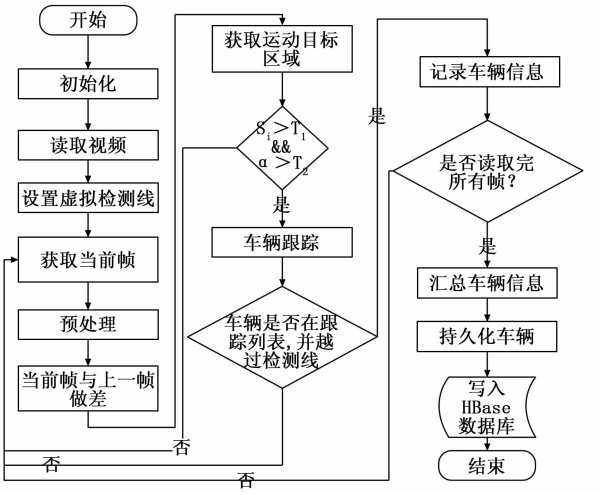

图 3 车流量检测算法流程图

分布式车流量检测方法软件编程主要基于 Java 语言，并配合开源分布式计算框架 Spark 和开源视频图像库 Java CV、FFmpeg。其中，视频文件编解码使用 FFmpeg 开源库实现；帧差法的车辆检测、跟踪与统计使用 Java CV 开源视频图像处理库实现；分布式数据获取、任务分配、车流量检测与检测结果汇总使用 Spark 框架实现；车辆信息持久化存储与获取使用 HBase API 实现。

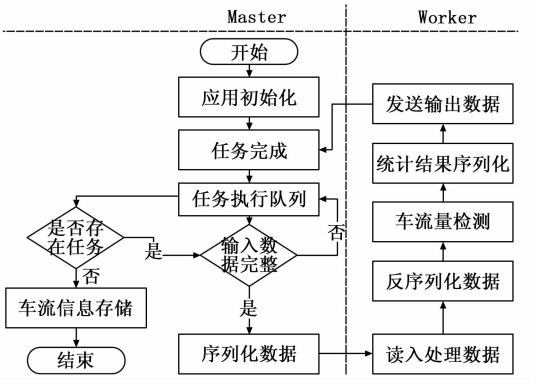

图 4 分布式车流量检测任务执行流程图

2.1 分布式视频获取

待处理的视频文件存储在 HDFS 中，Spark 不能对视频实现简单的分割处理。因此，本文采用两种方案完成视频文件的

获取，第一种为读取批量视频文件，每个计算任务处理一个完整的视频文件；另一种为读取批量视频，并采用视频帧分割方法完成视频读取，每个计算任务处理一定量的视频帧图像数据集合。

根据两种视频读取方案设计出两种分布式检测算法：基于视频帧分割的 Spark 分布式车流量检测算法（FBSVD，Frame－Based Spark Vehicle Detector）和基于完整视频处理的 Spark 分布式车流量检测算法（VBSVD，Video－Based Spark Vehicle Detector）。

第一种视频读取方案，只需获取 HDFS 中视频文件映射到本地的绝对路径集合，并将视频路径信息发送给计算节点。

第二种视频读取方案，必须切割视频帧图像数据。首先，传入任务总数，其中任务总数根据分布式集群中计算节点的 CPU 总核数进行设定，经验表明任务总数适宜为集群中 CPU 总核数的 2－3 倍；然后，根据任务总数计算出各节点需要获取到的视频帧区间；最后，计算节点在执行任务时解码视频获得需要处理的视频帧图像区间集合。具体操作伪代码如下：

```
Begin
Input 任务数量 N
获取待处理视频总帧数 n；
根据 N 计算每个任务需处理的视频帧数 g；
根据任务 id 获取本任务需要读取的视频开始帧 s；
if s+g>n then
本任务需要读取的结束帧 g=s+g；
else
本任务需要读取的结束帧 g=n；
Output 处理帧图像集合
End
```

2.2　图像预处理与车辆检测

本文通过使用图像预处理技术做基础操作，并使用帧差法检测运动目标。具体实现步骤如下：

1）预处理：使用图像二值化、高斯滤波、膨胀和腐蚀等方式完成待处理的两帧图像的预处理操作；

2）车辆检测：使用第 a 帧和第 $a-1$ 灰度图像作差，根据差图像判断车辆目标的存在。

两帧差分法检测运动目标公式如下：

$$D=|I(x,y,a)-I(x,y,a-1)| \tag{1}$$

式中，$I(x, y, a)$，$I(x, y, a-1)$ 分别表示第 a 帧和第 $a-1$ 帧的灰度图像，D 为差分结果图像。

检测出运动目标之后，对检测出的所有运动目标区域的面积记为 S_i，目标区域的宽高比记为 a，然后使用阈值检测的方式，判断运动目标区域是否为车辆记为 h，判断公式如下：

$$h=\begin{cases}1 & S_i\geqslant T_1, T_3>\alpha>T_2\\ 0 & \text{其他}\end{cases} \tag{2}$$

式中，T_1，T_2 和 T_3 为经验值，本文分别取为 5000，2 和 0.5。

2.3　车流量跟踪与统计

车流量跟踪与统计实现步骤如下：

1）虚拟检测线设置：实验设定为检测单向车道车流量，经过多次测试总结，检测线设置为视频高度的 0.54 倍，宽度的 0.5 倍；

2）车辆跟踪：将检测到的车辆加入待定车辆集合，并预计每个车辆在下一帧出现的可能方位。在检测到当前帧出现的车辆目标时，判断当前车辆的位移是否在待定车辆集合中某一车辆目标的合理位移范围内，从而实现目标跟踪；

3）车辆统计：当跟踪的目标经过了预设的检测线时，取消目标跟踪，并将车辆数加 1。

2.4　车流量信息持久化

通过车流量统计检测到的车辆信息，汇总检测视频中的车流量数量，以及检测的各车量信息，包括：车辆出现在视频中的哪一帧，以及在二维图像中的 x，y 坐标等，写入 HBase 分布式数据库。

2.5　分布式车流量检测算法实现

算法实现按照分布式集群节点类型，分为 Master 节点主控功能和 Worker 节点任务计算功能。其中，Master 节点作用为任务初始化、任务发起和任务结果回收入库的功能，Worker 节点作用为任务接收、任务执行和任务结果回送的功能。具体实现的核心伪代码描述如下：

```
Begin
Input V1, V2, …, Vk
Master 节点：

初始化应用。序列化待处理视频存储位置，并将其发送给计算节点；

Worker 节点：
反序列化任务处理视频 Vi 位置；
获取视频 Vi 的帧数 n；
for 1 to n
预处理相邻两帧图像；
相邻两帧的差帧 dFrame：dFrame=Is－In－1；
获取差帧中的运动区域集合 M；
while M≠ φdo
  if Smi≥T1 &&α>T2 then
   if 检测该区域为已跟踪的车辆 then
  if 车辆经过了检测线 then
   c=c+1；
记录该车辆信息到车辆集合 N；
Output 所有视频的车流量统计结果 N’
End
```

3　实验结果与分析

为了检验本文提出的 VBSVD 算法的性能和有效性，实验对两个交通路段的批量 5 min 交通视频进行测试，数据集名称和包含真实车数如表 1 所示（路段 A 的测试集记为 A－x，路段 B 的测试集记为 B－x）。设计了三组实验评价 VBSVD：1）与本文提出的 FBSVD 算法和帧差法之间在测试集 A－4 和 B－4 的比较；2）VBSVD 在不同测试集的比较；3）VBSVD 算

表 1　测试集信息表

视频编号	视频数量/个	实际车辆/辆	视频编号	视频数量/个	实际车辆/辆
A－1	10	122	B－1	10	299
A－2	12	151	B－2	12	364
A－3	20	260	B－3	20	632
A－4	24	336	B－4	24	726

法使用不同 CPU 核数测试 B－4，相对于传统帧差法测试 B－4 的比较。

实验采用普通四台 PC 机作为实验测试的分布式集群平台，其中三台计算节点（Inter 4 核，8 GB 内存，CentOS 7 操作系统），一台主控节点（Inter 2 核，2 GB 内存，CentOS 7 操作系统）。

表 2 展示了本文提出的 FBSVD、VBSVD 以及传统帧差法针对 A－4 的准确率和检测时间结果，图 5 展示了 3 种检测算法在路段 A 的不同数据集的处理时间对比。结果表明，VBSVD 算法与传统帧差法具有相同的检测准确率，VBSVD 算法比 FBSVD 算法准确率高 3.3%；VBSVD 算法比传统帧差法检测速度快 510%，传统帧差法比 FBSVD 快 252%。

表 2 传统帧差法、FBSVD、VBCVD 测试 A－4 的结果表

算法	检测车辆/辆	准确率/%	时间/s
帧差法	293	87.2	2456.6
FBSVD	282	83.9	6203.7
VBSVD	293	87.2	481.6

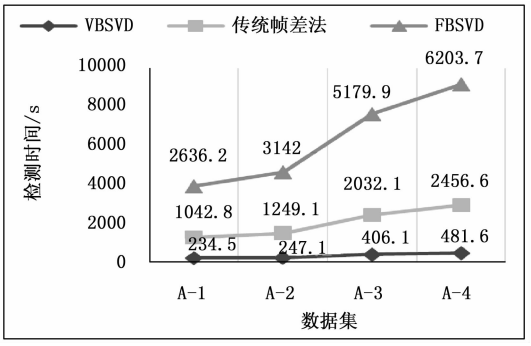

图 5 传统帧差法、FBSVD、VBCVD 测试路段 A 的不同数据集的时间对比

表 3 传统帧差法、FBSVD、VBCVD 测试 B－4 的结果表

算法	检测车辆/辆	准确率/%	时间/s
帧差法	687	94.6	2528.9
FBSVD	672	92.5	6329.3
VBSVD	687	94.6	462.3

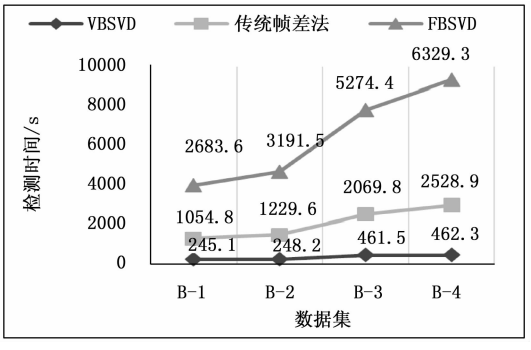

图 6 传统帧差法、FBSVD、VBCVD 测试路段 B 的不同数据集的时间对比

表 3 展示了本文提出的 FBSVD、VBSVD 以及传统帧差法针对 B－4 的准确率和检测时间结果。对于路段 B 的不同数据集，3 种检测算法的处理时间对比如图 6 所示。结果表明，传统帧差法与 VBSVD 算法具有相同的检测准确率，传统帧差法比 FBSVD 准确率高 0.9%；VBSVD 算法比传统帧差法检测速度快 547%，传统帧差法比 FBSVD 快 251%。

表 4 VBSVD 算法测试路段 A 的不同测试集的结果表

数据集	检测车辆/辆	准确率/%	运行时间/s
A－1	107	87.7	234.5
A－2	131	86.7	241.7
A－3	226	86.9	406.1
A－4	293	87.2	481.6

表 5 VBSVD 算法测试路段 B 的不同测试机的结果表

数据集	检测车辆/辆	准确率/%	运行时间/s
B－1	275	91.9	245.1
B－2	339	93.1	248.2
B－3	587	92.8	461.5
B－4	687	94.6	462.3

表 5 展示了 VBSVD 算法针对路段 A 的不同测试数据集的检测时间和检测准确率结果。表 6 展示了本文提出的 VBSVD 针对路段 B 的不同测试数据集的检测时间和检测准确率结果。

图 7 展示了对于数据集 B－4，在不同 CPU 使用数量的情况下，VBSVD 算法相对于传统帧差法的处理加速比。

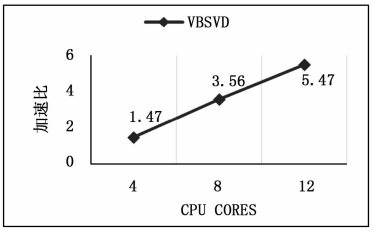

图 7 VBSVD 针对不同数据集 B－4 使用在不同 CPU 核数情况下的加速比

实验结果表明，本文提出的 VBSVD 算法比传统帧差法检测速度平均快 528%，主要原因是：VBSVD 算法充分利用集群中的计算节点的 CPU 和内存资源，采用分布式多路检测的方式提升检测速度。传统帧差法比 FBSVD 算法比快 251%，主要原因是：FBSVD 算法对视频帧进行切割，导致了时间消耗。

实验结果分析证明：视频文件具有不易切分的特点，宜将视频整体处理；VBSVD 算法使用的分布式集群中 CPU 数量越多提升的检测速度越明显，当 CPU 核的数量为待处理数据集的 1/3 或 1/2 时，效果最好。

4 结束语

针对大规模交通视频下的批处理车流量检测存在计算开销大以及检测时间过长的问题，本文通过对 Apache Spark 的研究，并将 Spark 应用到车流量检测当中，实现了批量交通视频的车流量检测并行化。相对于传统检测方法，本文提出的 VBSVD 算法能依靠分布式集群的计算节点同时处理批量交通视频，完成检测任务。经实验表明，VBSVD 算法简单可行，准确率高，检测速度快，为检测批量交通视频提供了较好的解决方案。

（下转第 206 页）