

一种应用于边缘计算框架的 改进型动态目标跟踪方法

刘 浩^{1,4}, 卿 粼波², 宗江琴³, 陈虹君¹

(1. 成都锦城学院电子信息学院, 成都 611731; 2. 四川大学电子信息学院, 成都 610065;
3. 江西省信息科技学校, 南昌 330006; 4. 成都锦城学院四川省专家工作站, 成都 611731)

摘 要: 随着 5G 通信和物联网大数据技术的高速发展, 传统的云计算模式已经越来越跟不上数据的增长速度了, 边缘计算作为一种新的计算模式, 表现出了很强的处理大数据和高速计算的能力. 本文在“智能交通仿真系统”课题中, 提出了一种适合于视频图像处理的边缘计算框架, 并对传统的动目标跟踪算法进行了两点改进: (1) 采用树莓派作为视频前端处理器, 具有体积小, 成本低, 计算能力强的特点, 适合边缘计算; (2) 针对跟踪阶段需要采集大量样本, 计算量大的缺点, 采用了一种较小步长作为滑动窗的分步式图像采样方法对原压缩跟踪算法进行改进, 从而减少了计算量. 计算机仿真实验的结果证实了该算法在基本不影响跟踪精度的情况下提高了运算速度.

关键词: 边缘计算; 树莓派; 稀疏表示; 压缩跟踪; 分步式采样

中图分类号: TP751.1 **文献标识码:** A **DOI:** 10.19907/j.0490-6756.2022.063003

An improved moving target tracking method suitable for edge computing framework

LIU Hao^{1,4}, QING Lin-Bo², ZONG Jiang-Qin³, CHEN Hong-Jun¹

(1. College of Electronic Information Engineering, Chengdu Jincheng College, Chengdu 611731, China;
2. College of Electronic Information Engineering, Sichuan University, Chengdu 610065, China;
3. Jiangxi Information Technology School, Nanchang 330006, China;
4. Sichuan Expert Workstation of Chengdu Jincheng University, Chengdu 611731, China)

Abstract: With the rapid development of 5G communications technology, IoT and big data technology, traditional cloud computing models have become increasingly unable to keep up with the growth rate of data, as a new computing model, edge computing has demonstrated a strong ability to handle big data and high-speed computing. This paper propose an edge computing framework suitable for video image processing, and two improvements to the traditional moving target tracking algorithm: (1) Raspberry Pi is used as the video front-end processor, which has the characteristics of small size, low cost, and strong computing power; (2) a step-by-step image sampling method with a smaller step size as a sliding window is used to improve the original compression tracking algorithm, thereby reducing the amount of calculation. The results of computer simulation experiments show that the algorithm improves the operation speed without affecting the tracking accuracy.

Keywords: Edge computing; Raspberry; Sparse representation; Compression tracking; Stepwise sampling

收稿日期: 2022-01-17

基金项目: 四川省科技厅重点研发项目(22ZDYF0724)

作者简介: 刘浩(1969—), 男, 硕士, 副教授, 主要研究方向为信号处理. E-mail: liuh6679@163.com

通讯作者: 卿粼波. E-mail: qing_lb@scu.edu.cn

1 引言

从 2019 年开始,全球 5G 通信得到了高速发展,商用规模越来越大,网络容量大大增加,很多得益于 5G 高速通信的新技术和新应用领域扩张极快,例如:智慧城市,智能工厂和物联网等.其中,边缘计算是发展潜力最好的方向之一^[1].

边缘计算(Edge Computing)是一种新的网络计算模型,它采用网络、计算、存储和应用核心能力为一体的开放平台,就近提供近端服务,边缘计算,将原有基于云计算的全部或部分计算任务迁移到网络边缘设备上,其应用程序在边缘侧发起,产生更快的网络服务响应,这意味着许多数据处理和计算将通过本地设备实现,而无需交由云端,处理过程将在本地边缘计算层完成,这无疑将大大提升处理效率,减轻云端的负荷^[2].

对于智能交通这种要求短时延、高带宽、高可靠、海量连接网络通信的场合,边缘计算相对于传统的云计算有很大的优势^[3].

现在的智能交通系统,前端有海量的视频数据,同时系统对视频数据的处理速度,传输时延等提出了很高的要求,这样才能保证视频流的实时性.基于这个目标,本文利用实验室中现有的“智能交通仿真系统”,用一种边缘计算框架来改进此系统中的视频采集及传输模块,并提出了一种适合于这种架构的目标跟踪方法对系统中的运动目标进行追踪.

本“智能交通仿真系统”包含城市道路、交通标志、隧道、ETC 收费站、智能停车场和红绿灯路口等场景要素,如图 1 所示.



图 1 智能交通仿真系统
Fig. 1 Intelligent transportation simulation system

2 边缘计算框架

2.1 边缘计算 Cloudlet 架构

Cloudlet 是 2013 年由卡内基梅隆大学提出

的概念,它将原来移动计算的两层架构“移动设备-云”改进成三层架构“移动设备-cloudlet-云”^[4].

其中,Cloudlet 作为中间层,部署在网络边缘,自带硬件组件及操作系统,负责对前端(移动设备)的数据进行采集,分析和处理,并将结果上传后方云中心.

2.2 边缘计算视频处理框架

基于 Cloudlet 架构的基本原理,本文提出了一种适合于实验室具体情况的边缘计算视频处理框架.

视频处理框架主要由边缘节点 ECN(Edge Computing Node)和边缘节点管理器(ECN Controller)组成,ECN 负责对前端采集到的视频信号进行处理,包括视频跟踪和目标识别等,边缘节点管理器负责为 ECN 动态分配资源,把各路 ECN 上传的视频处理结果传到远端的云中心进一步处理,如图 2 所示.

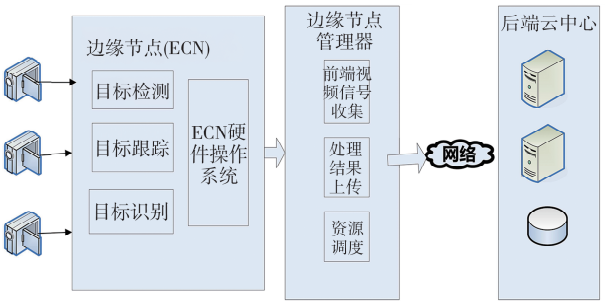


图 2 边缘计算视频处理框架
Fig. 2 Video processing framework for edge computing

在以边缘计算为基础的智能交通系统中,边缘节点(ECN)主要由带处理器的各种传感器(摄像头,雷达,...)组成,它对采集到的视频信号预处理,然后把处理结果上传到边缘节点管理器.实际的交通系统中有各种传感器,以各种不同的通信方式上传信息,例如,无线 WIFI 方式,蓝牙方式和 ZigBee 方式等.边缘节点管理器接收边缘节点传来的处理过的视频流,向上转发到云中心,且把云中心的指令下发给各边缘节点,边缘节点管理器通过互联网接入云中心,接入方式有 HTTP 协议,Web 服务中的 Rest 方式,物联网的 MQTT 协议等,如表 1 所示.

本实验就是利用现有实验室“智能交通仿真系统”来模拟一个基于上述视频处理框架的边缘计算系统,阿里云或百度云等平台作为后端云中心,基于 Cortex-A9 的工控机作为边缘节点管理器,负责接收边缘节点发来的视频处理结果并上传至云中心.

表 1 边缘计算框架中的通信协议

Tab. 1 Communication protocols in edge computing framework

模块	通信协议(接入方式)
边缘节点(ECN)~ 边缘节点管理器	WiFi (802. 11), Zigbee 协议 (802. 15), 蓝牙(L2CAP, LMP), 局域网协议(TCP/IP, IPX/SPX)...
边缘节点管理器(ECN controller)~ 云中心	HTTP, TCP/IP, MQTT Rest(Web Service)...

按照边缘计算的原理和要求,实时视频图像分析和处理必须在边缘节点完成,并把结果上传到边缘节点管理器,这就要求边缘节点需要有较强的视频处理功能,但又不能有过的体积和重量,为此,本文采用树莓派 3(Raspberry Pi 3)作为“智能交通仿真系统”的边缘节点。

树莓派是非常廉价的、信用卡体积的世界上最小的卡片式电脑,树莓派 3(Raspberry Pi 3)基于 Linux 操作系统,采用了 64 位处理器:主频 1. 2 GHz,拥有 1 GB RAM 和 VideoCore IV GPU,有较强的视频编解码能力,能够流畅地处理视频和图像^[5]。

在本系统中,树莓派作为边缘节点,主要功能是接收前端摄像头传来的视频信号,对系统中的运动小车进行实时跟踪。

3 压缩跟踪算法

现代运动目标跟踪的主要方法有:基于纹理特征的跟踪^[6,7],边缘特征的跟踪^[8],光流特征的跟踪^[9]等。在跟踪算法方面,有对样本要求较少 KCF 跟踪算法^[10-12],还有跟踪速度较快,而且能部分解决目标遮挡问题的粒子滤波法^[13]。

上述几种跟踪算法具有跟踪精度高,鲁棒性好的优点,但也存在学习样本较多,计算量较大的不足,因为树莓派处理器的运算能力还是无法和基于 inter 芯片的服务器(现今主流云计算配置)相比,因此不是很适合那几种较为复杂的算法。2012 年, Zhang 等^[14]提出了一种基于压缩传感的动目标跟踪算法,此算法建立在信号稀疏表示基础上,首先使用稀疏测量矩阵提取目标和背景的特征,作为在线学习更新朴素贝叶斯分类器的正样本和负样本,然后使用更新后的分类器去分类下一帧图像中的目标与背景样本,实现动目标实时跟踪,具有算法简单,高效实时的特点,本文根据实际情况对此算法做了一定的改进,使之更适合在树莓派上运行,实验证实了改进的有效性。

3.1 压缩特征提取

按照压缩传感理论,只要信号 X 是 K -稀疏的,一个图像空间的高维信号 $X \in R^n$ 可以通过一个随机矩阵 $A \in R^{m \times n} (m \ll n)$,投影到低维空间 $A \in R^m$,如图 3 所示。

$$Y = Ax \tag{1}$$

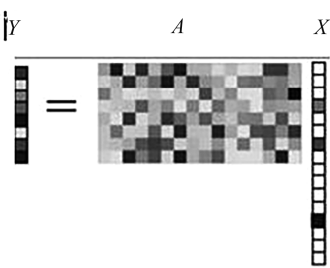


图 3 投影矩阵对图像信号降维
Fig. 3 Dimension reduction of image signal by projection matrix

图 3 中, X 是 N 维列向量,代表输入高维信号, X 是 K -稀疏的, A 是 $M \times N$ 维测量矩阵,输出 Y 是 M 维列向量。式(2)本质是利用测量矩阵对高维图像特征进行降维。

由压缩传感理论可知,只要 K -稀疏信号 X 满足约束等距 RIP(Restricted Isometry Property)准则,就能够使 N 维向量 X 到 M 维向量 Y 的降维过程中,信号 X 的信息不被破坏,也就是说投影后的低维子空间可以很好的保留高维图像特征空间的信息。

3.2 跟踪步骤

(1) 使用测量矩阵降维。

按照压缩传感跟踪理论^[14],在第 $t-1$ 帧图像中,在已确定的跟踪目标位置四周,以 $D^* = \{z | \|I_z - I_t\| < \alpha\}$ 为半径的区域内采取正负目标样本图像片,用符合 RIP 条件的测量矩阵按照式(1)对目标样本图片降维,得到每个图像片的特征向量 Y ,测量矩阵 A 采用如下的矩阵元素。

$$r = \sqrt{s} \times \begin{cases} 1, & p = 1/2s \\ 0, & p = 1 - 1/s \\ -1, & p = 1/2s \end{cases} \tag{2}$$

其中, r 是测量矩阵 A 的元素; P 代表 r 取某值的概率;算法中设定 $S = M/4$; M 是上述矩阵维度。

针对图像信号,采用的是 Haar-like 特征作为压缩特征,稀疏降维后的特征向量 Y 中的每个元素都是不同尺度下 Harr-like 特征的线性表示,保留着原始目标的图像特征信息。

(2) 朴素贝叶斯分类器的创建和更新。

对于上述目标样本的特征向量 Y , 可以用贝叶斯分类器来进行分类, 分类器如式(3).

$$H(V) = \ln \left[\frac{\prod_{i=1}^n p(v_i | y = 1) p(y = 1)}{\prod_{i=1}^n p(v_i | y = 0) p(y = 0)} \right] = \sum_{i=1}^n \ln \left(\frac{p(v_i | y = 1)}{p(v_i | y = 0)} \right) \quad (3)$$

用上式贝叶斯分类器对特征向量 Y 进行分类, 通过计算最大分类器响应的样本作为第 t 帧里的目标的位置 I , 用步骤(1)同样的方法采集两个样本集合, 一个是离目标近的正样本, $D^+ = \{z | \|I_z - I_t\| < \alpha\}$; 一个是离目标较远的负样本, $D^- = \{z | \eta < \|I_z - I_t\| < \beta\}$, 其中 $\alpha < \eta < \beta$.

(3) 重复步骤(1), 从上述正负样本中提取特征, 送到贝叶斯分类器中学习, 然后更新分类器参数.

(4) 重复上述步骤, 实现运动目标跟踪.

4 对压缩跟踪算法的改进

压缩跟踪算法具有算法简单, 鲁棒性好的特点, 但是跟踪速度不是很快, 这是因为该算法在按照上述 2.3 节的步骤搜索目标窗口, 采集样本方式是: 在原图像中设定一个比较大采样半径 $D^+ = \{z | \|I_z - I_t\| < \alpha\}$, 在此区域内采用滑动窗法(就是一个像素一个像素逐一采样的办法)采取正负目标样本图像片, 得到样本后, 用测量矩阵降维得到目标特征. 因为是滑动窗逐一采样, 目标图像的像素很大, 而且一般按每个样本 50 个目标特征来计算, 这样的搜索采样方式计算量很大, 导致跟踪速度偏慢.

本文对上述算法进行改进, 提出了一种分步搜索获取采样图像样本的方法, 如图 4 所示, 具体步骤如下.

(1) 对视频 $t-1$ 帧, 按照压缩跟踪算法, 在 $D^+ = \{z | \|I_z - I_t\| < \alpha\}$ (例如: 以半径 20 像素为正负样本的分界)区域内, 不再用滑动窗法逐一像素地采集图像样本, 而是分为以下两步.

第一步: 以当前目标位置为中心, 以 20 像素为半径的一个相对较大的范围内以 4 像素为步长进行滑动搜索, 就是说每 4 个像素点采集一次样本, 如图 5 所示. 滑动窗从起始点 A 出发, 然后在半径 20 像素的范围内, 以 4 像素为步长, 移动采样, 图中黑色圆形是以 A 点出发的第一次可能到达的 8 个点位置, 然后按照算法进行降维, 提取图像特征,

送入贝叶斯分类器分类, 并计算相应的 $H(V)$, 取最大 $H(V)$ 所对应的目标位置作为当前最佳目标位置.

第二步: 从上一步取得的最佳位置(假设图中 B)开始, 以半径为 10 的搜索范围内, 以 2 为步长进行滑动窗搜索采样, 图中黑色菱形点就是以 B 点出发的第二次可能到达的 8 个点位置, 然后提取图像特征, 比较每一个位置的 $H(V)$, 最大 $H(V)$ 所对应的目标位置为最终的目标位置.

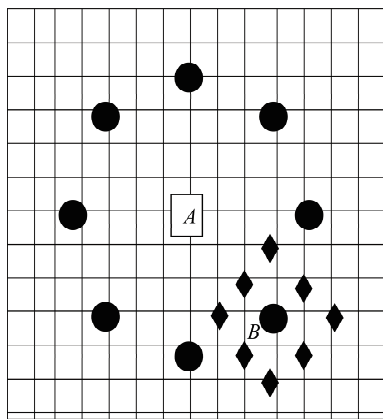


图 4 改进压缩跟踪算法的分步式搜索示意图

Fig. 4 Schematic diagram of step search for improved compression tracking algorithm

(2) 重复上述步骤, 实现运动目标跟踪.

如图 4 所示, 按照压缩跟踪算法^[15], 滑动窗搜索法采集的样本数是 $Sample = \frac{\pi r^2}{(\Delta f)^2}$, 其中 r 是搜索半径, Δf 是搜索步长. 采用全局搜索算法时, 半径 $r=20$, $\Delta f=1$, 算得样本数 $Sample$ 大约为 1256; 采用本文的分布式采样法, 样本数 $Sample$ 大约为 157(即 $S = \frac{\pi \times 20^2}{4^2} + \frac{\pi \times 10^2}{2^2}$).

算法总计算次数计算公式如下式.

$$S = Sample + Number + Count \quad (4)$$

其中, S 是算法总计算次数; $Sample$ 是采集的样本数(依赖于搜索窗口的大小), $Number$ 是各样本的特征数目(一般设定为 50), $Count$ 是各特征的计算次数(一般设定为 2~4 次).

在各样本的特征数目和各特征的计算次数一致的情况下, 滑动窗搜索法采集的样本数越少, 跟踪速度越快, 从上述分析可看出, 本文的分步式采样能提高算法的跟踪速度.

5 实验结果

实验中, 采用第 2 节所述的带摄像头的树莓派

作为视频前端处理器,用 C++ 语言结合 OpenCV 视觉库开发目标跟踪程序.

为了验证这种算法,本文用公开数据集中的 David, bolt 视频以及本“智能交通仿真系统”中的运动小车视频作测试比较.

公开数据集中的 David 视频,测试效果如图 5.

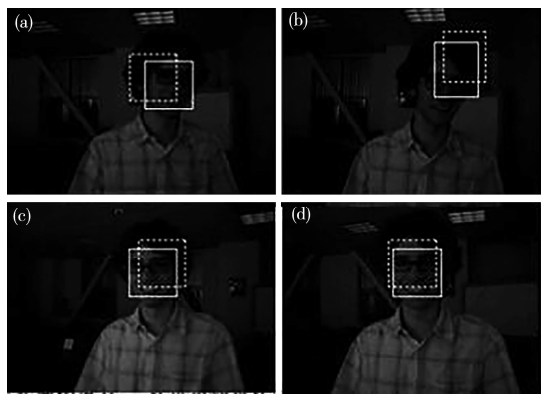


图 5 两种算法对 David 视频实时跟踪效果
Fig. 5 Effect of two algorithms on David video

图 5 中,(a)~(d)4 幅图分别是第 25 帧,56 帧,120 帧和 175 帧视频,实线框是原始压缩跟踪算法跟踪结果,虚线框是本文改进算法跟踪结果,从图 5 看,都实现了良好跟踪.

本文使用目标跟踪理论常用评价标准中的中心位置误差(Center Location Error, CLE)来评价算法的跟踪精度.

中心位置误差(CLE)定义预测的目标中心和真实的目标中心之间的欧氏距离,以像素为单位.

$$CLE = \sqrt{(x_a^2 - x_b^2) + (y_a^2 - y_b^2)}$$

(5)

其中,(x_a, x_b)表示 Ground truth 中预测目标中心位置,(y_a, y_b)表示真实的目标中心位置.

在实际视频应用中,容易测到的是目标的左上角坐标(x, y)及跟踪矩形长宽 ($width, height$)

它们和中心位置之间的转换如下.

$$x_a = x + width / 2$$

$$y_a = y + height / 2$$

对 David 视频跟踪误差图如图 6 所示.

从图 6 可看出,原始压缩跟踪算法和本文的改进算法的中心位置误差相差不大.对公共数据集中的 bolt 视频,测试效果如图 7.

图 7 中,(a)~(c)3 幅图分别是第 40 帧,160 帧和 235 帧视频,可以看出,两种算法都是实时跟踪,因为跑步速度较快,在最后稍微有点“丢帧”现象,跟踪误差图如图 8. 到视频后期(250 帧)以后,

误差稍变大.

本文重点工作是针对实验室“智能交通仿真系统”,把运动小车作为跟踪目标,分别用压缩跟踪算法和改进型算法进行跟踪,实时跟踪效果如图 9.

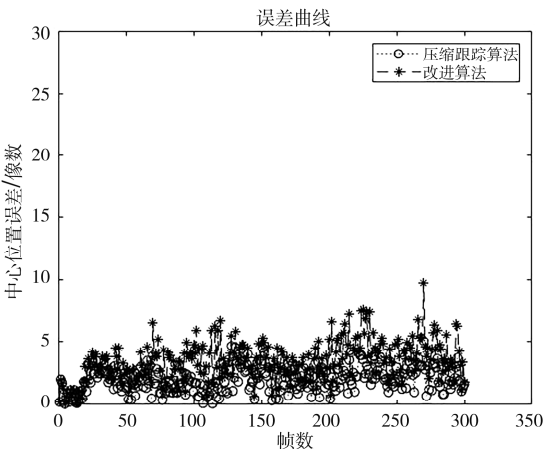


图 6 David 视频序列跟踪的误差曲线图
Fig. 6 Error curve of David tracking video sequence

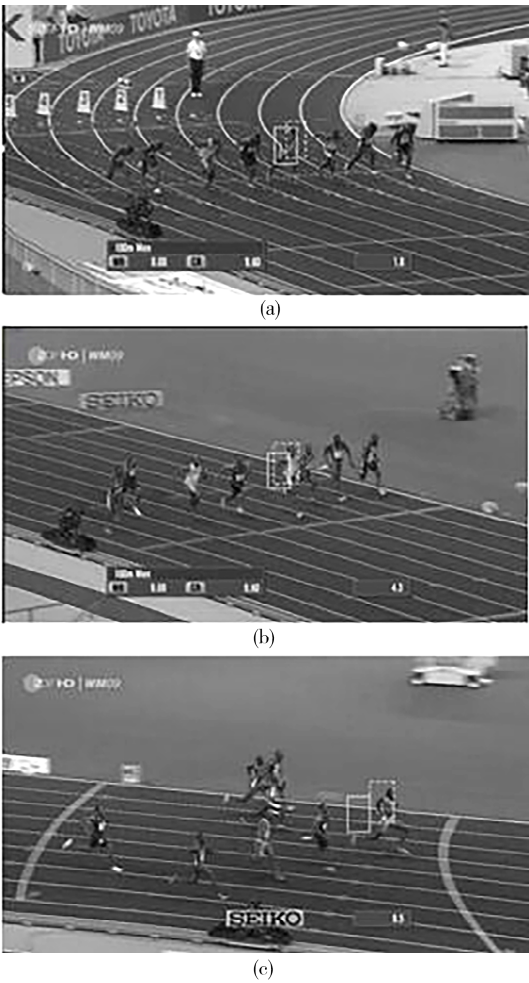


图 7 两种算法对 bolt 视频实时跟踪效果
Fig. 7 Real-time tracking effect of two algorithms on bolt video

图 9 是压缩跟踪算法和改进型算法对智能小车实时跟踪的截图,(a) ~ (d) 分别是第 16 帧,65 帧,155 帧和 180 帧的跟踪图,实线框是原始压缩跟踪算法跟踪结果,虚线框是本文改进算法跟踪结果因为小车运动速度不算太快,两种算法都能实现 100%跟踪,无丢帧现象。

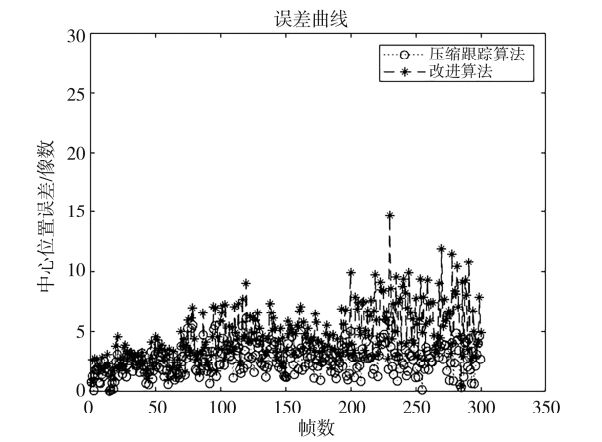


图 8 bolt 视频序列跟踪的误差曲线图
Fig. 8 Error curve of bolt tracking video sequence

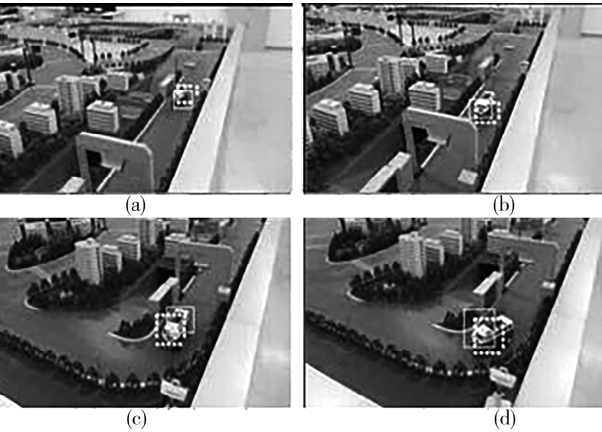


图 9 两种算法对智能小车实时跟踪效果
Fig. 9 Real time tracking effect of two algorithms on intelligent vehicle

从文献[15]中,得知压缩跟踪算法跟踪速度可以达到接近 40 帧/s,可能因为操作系统或者处理器硬件的关系,实际测试中达不到这个速度。

实验中进行跟踪速度对比测试,除了原始压缩跟踪算法和本文的改进算法外,还用目标跟踪算法中常用的光流法作对比,三种算法对沙盘小车跟踪的实时比较如表 2,其中 CT 表示传统的压缩跟踪算法。

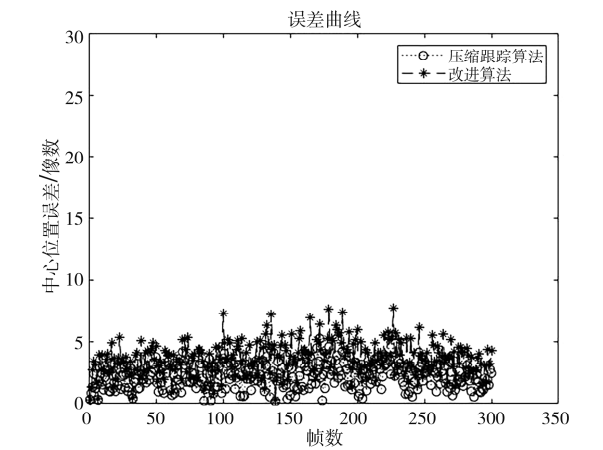


图 10 运动小车视频序列跟踪的误差曲线图
Fig. 10 Error curve of car tracking video sequence

表 2 压缩跟踪算法和改进型算法跟踪速度对比
Tab. 2 Comparison of tracking speed between compression tracking algorithm and improved algorithm

算法	帧数	处理时间/s	帧率/ $f \cdot s^{-1}$
光流法	428	15.5	16.7
CT	428	20.4	20.9
改进型 CT	428	13.8	30.9

6 结 论

本文方法是对压缩跟踪算法的一个改进,采用了一种较小步长作为滑动窗的分步式图像采样方法来代替原算法中逐点采样的方法,从而减少了计算量,实验证明了这种方法的有效性。

参考文献:

[1] 施巍松, 孙辉, 曹杰. 边缘计算: 万物互联时代新型计算模型[J]. 计算机研究与发展, 2017, 54: 907.

[2] 吴肖伶, 张记会, 龚星衡, 等. 一种基于边缘计算的 RTK 定位方法[J]. 四川大学学报: 自然科学版, 2021, 58: 012002.

[3] Pan J, Mcelhannon J. Future edge cloud and edge computing for internet of things applications [J]. IEEE Internet of Things J, 2017, 5: 439.

[4] Satyannarayanan M, Bahl P, Caceres R, et al. The case for VM-based cloudlets in mobile computing [J]. IEEE PervasComput, 2009, 8: 14.

[5] Schmidt M, Raspherry P. 快速入门指南[M]. 王峰, 等译. 北京: 科学出版社, 2014.

[6] 孙君顶, 马媛媛. 纹理特征研究综述[J]. 计算机系统应用, 2010, 19: 245.

[7] 谭翔纬. 基于支持向量机和用户反馈的图像检索算

法[J]. 吉林大学学报: 理学版, 2020, 58: 899.

[8] Myung-Cheol R, Tae-Yong K, Park J H. Accurate object contour tracking based on boundary edge selection [J]. Pattern Recogn, 2007, 3: 931.

[9] Kristan M, Per J, Kovai S, *et al.* A local-motion-based probabilistic model for visual tracking [J]. Pattern Recogn, 2009, 9: 2160.

[10] 周正松, 陈虹君, 周红. 基于多特征融合的尺度自适应 KCF 目标跟踪算法[J]. 四川大学学报: 自然科学版, 2020, 57: 697.

[11] 宫金良, 陈涛, 张彦斐, 等. 一种基于多区域信息融合约束的改进帧间差分目标检测与跟踪算法[J]. 江苏大学学报: 自然科学版, 2022, 43: 302.

[12] 赵蕾, 谢鹏. 基于多视角特征协同融合的红外导引头目标追踪算法[J]. 重庆邮电大学学报: 自然科学版, 2020, 32: 639.

[13] Wu Y, Lim J, Yang M H. Online object tracking: A benchmark [C]//Proceedings of the 2013 IEEE Conference on Computer Vision and Pattern Recognition (CVPR). Portland, Oregon: IEEE, 2013.

[14] Zhang K H, Zhang L, Yang M H. Real-time compressive tracking [C]//Proceedings of the European Conference on Computer Vision. Firenze, Italy: Springer, 2012.

[15] Zhang K H, Zhang L, Yang M H. Fast compressive Tracking [J]. IEEE T Pattern Anal, 2014, 36: 2002.

引用本文格式:

中 文: 刘浩, 卿鄰波, 宗江琴, 等. 一种应用于边缘计算框架的改进型动态目标跟踪方法[J]. 四川大学学报: 自然科学版, 2022, 59: 063003.

英 文: Liu H, Qing L B, Zong J Q, *et al.* An improved moving target tracking method suitable for edge computing framework [J]. J Sichuan Univ: Nat Sci Ed, 2022, 59: 063003.