An Initial Study of Real-time Traffic Estimation with Low-Cost IoT Cameras and OGC IoT Standards

Sepehr Honarparvar[1], Sara Saeedi[2,3], Steve Liang[1,2]

[1]Department of Geomatics Engineering, University of Calgary,{Sepehr.honarparvar, ssaeedi, Liangs }@ucalgary.ca
[2] SensorUp Inc., steve.liang@sensorup.com
[3] Open Geospatial Consortium, ssaeedi@ogc.com

ABSTRACT

Real-time and automated traffic flow estimation is an important issue in growing cities. A low-cost real-time road traffic estimation method leveraging low-cost off-the-shelf Internet of Things (IoT) edge devices could be scalable and economical to deploy. Recent advancements in computer vision and Deep Learning (DL) technologies offer the possibility to enable low-cost IoT cameras to automatically monitor and analyze objects' movements. In this short paper, we proposed a pub/sub architecture based on OGC IoT standards that automate and publish real-time traffic flow estimations from multiple distributed IoT edge devices. For the traffic flow estimation, the number of cars, the average speed during a certain time window, and the number of pedestrians are predicted. To evaluate the method the OGC Smart City Interoperability Reference Architecture (SCIRA) data is used. Based on the results, the precision of counting cars is 93%, the precision of counting pedestrians is 87%, the precision of the average speed estimation is 78%, and the performance of the architecture is 846 milliseconds for every traffic report.

1. Introduction

Urban development has brought economic benefits to people and governments. It caused urban road network development as well as the rising demand for personal vehicles. Traffic control has become a critical issue in urban areas because of the growing number of cars and the increasing demand for transportation (Papageorgiou et al, 2003). Traffic estimation is one of the most important tasks of a traffic control loop (Papageorgiou et al, 1983). Traffic estimation in urban roads refers to the measurement of traffic variables. Traffic variables include average car speed, the number of cars, and the density of cars in a road segment. These variables can be estimated from Closed Circuit Television (CCTV) using Deep Learning (DL) object detection solutions. Object detection should be accomplished frame by frame. To detect objects in each frame, raw frames can be sent to the cloud services. This approach has been discouraged because of bandwidth and latency challenges (Kar et al., 2017). Regarding recent advances in DL and image processing, edge computing techniques have been widely applied as a low-cost and fast solution for real-time traffic data processing (Barthélemy et al., 2019). However, assigning all traffic processes to the edge computing part is not efficient since, in some cases, a huge amount of memory should be dedicated to keeping frames observations for calculations. Therefore, a combination of cloud and edge computing architectures is required to process frames in real-time while providing efficient memory usage for edge computing devices. This paper proposes a pub/sub edge/cloud architecture to estimate traffic in real-time using CCTV cameras. One of the main challenges of this architecture is frame loss due to inefficient memory management on edge. As all frames should be detected in real-time, some frames should be skipped to capture and process the on-time frames. Losing frames would affect the object tracking as well as observation aggregations through the consecutive frames. This paper tries to demonstrate a method to reach a better accuracy of traffic estimations by assigning different parts of the processing tasks to the architecture nodes.

2. Methods & Data

The methodology and data section of this paper is organized into three sub-sections. In the first sub-section, the method of estimating traffic is elaborated. The second sub-section explains the proposed pub/sub architecture and the details of the different architecture nodes. In the third sub-section, the implementation and data are explained.

2.1. Traffic estimation method

To estimate traffic, three traffic parameters (i.e. the number of cars, the number of pedestrians, and the vehicle's speed) are calculated. Figure 1 illustrates how these parameters are calculated.

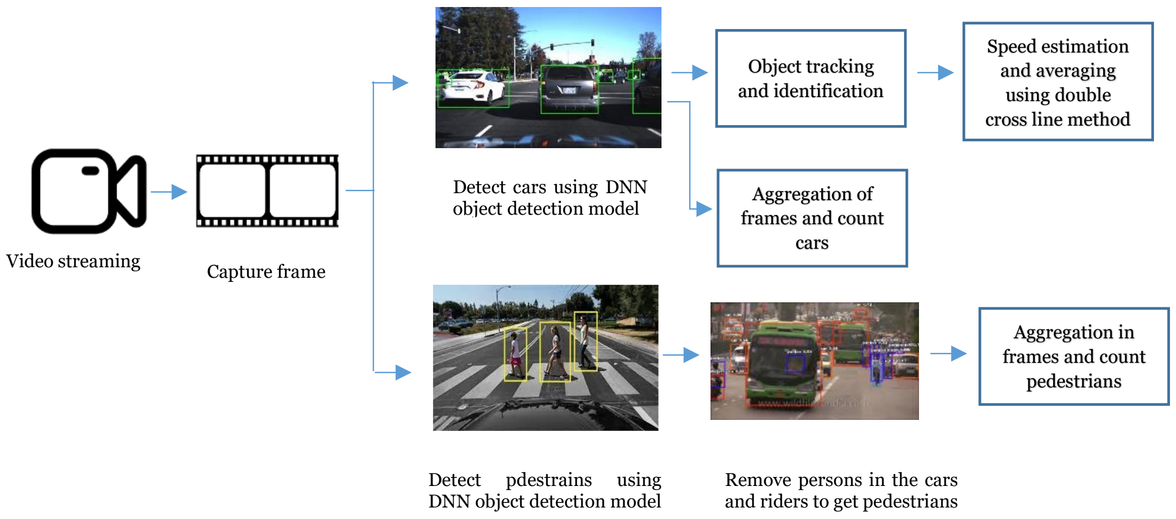
Video streaming
Capture frame
Detect cars using DNN object detection model
Object tracking and identification
Speed estimation and averaging using double cross line method
Aggregation of frames and count cars
Detect pdestrains using DNN object detection model
Remove persons in the cars and riders to get pedestrians
Aggregation in frames and count pedestrians

Figure 1- traffic estimation method

In this method, frames are captured from camera feeds. Then, cars and persons are detected using a Deep Neural Network (DNN) model e.g. You Only Look Once (Yolo) (Redmon and Farhadi, 2018). Yolo provides higher accuracy for moving object when high-speed object detection and tracking is required (Honarparvar et al., 2021). For pedestrians, person objects such as riders and drivers should be excluded from the person class list to have only pedestrians objects. Then using an aggregation function all pedestrians are counted and aggregated through frames. The same approach is applied to count and aggregate cars. To calculate cars' speed, a tracking function should be run on the detected cars. For tracking and identifying cars, Simple Online and Realtime Tracking (SORT) is used (Bewley et al., 2016). Then by projecting the car's bottom center to the geographic space, a trajectory on the geographic coordinate system is built. Having two parallel virtual lines on the street, the start and the end of the trajectory are determined. Consequently, the speed is estimated using the distance and the time window. The aggregated speed for all cars in a specific time window would be considered as the estimation of cars' speed. Figure 2 illustrates the workflow of estimating a car's speed. For projecting the points to the geographic coordinates system, we used the homography (projective) transformation and four ground control points (Honarparvar et al., 2021). Due to the trajectory incorrect perturbation, the trajectory length is often more than the real value. Therefore, a two-dimensional Kalman Filter is applied to smooth the path and reduce the trajectory outliers (Barrios, C., & Motai, 2011).

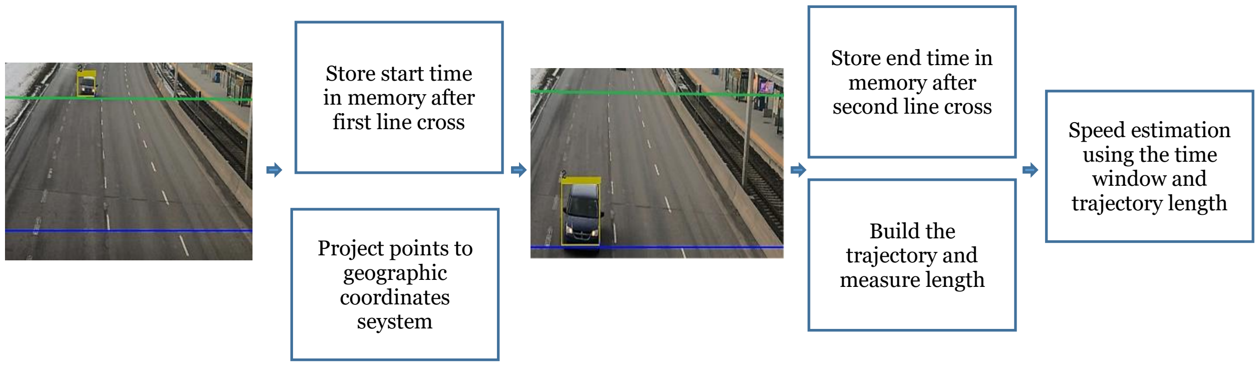
Store start time in memory after first line cross
Project points to geographic coordinates seystem
Store end time in memory after second line cross
Build the trajectory and measure length
Speed estimation using the time window and trajectory length

Figure 2- speed estimation workflow

2.2. The proposed pub/sub traffic estimation architecture

The proposed architecture is based on a pub/sub pattern. The pub/sub architecture is a suitable pattern for handling distributed sources and tasks and it will increase the scalability (Onica et al., 2016). In this architecture, camera feeds are processed by the edge computing node. Then, the frames are published to a message broker to distribute the messages to a proper subscriber for handling the traffic estimations tasks. Due to the limitation of the memory usage of the edge processors, the object detection and tracking processes are assigned to the edge computing part. The rest of the processes, including coordinates projection, trajectory refinement, data aggregation, and speed estimation are handled in the cloud computing part. Figure 3 illustrates the details of the architecture. In the architecture, feeds are processed by a trained DNN object detection model, then objects are sent into a tracking function to identify each object. After this step, a payload including the object's class, identification number, bounding box, the frame's capture time, and the camera ID is published to a message broker right after the tracking step. The message broker gets the payload and distributes them to subscribers including two engines and one database. The frame data aggregation engine gets the vehicle and pedestrians data and formats them into a partition key (i.e., object class) and sort key (i.e., capture time) to add new rows to the temporary observations table. The aggregation engine queries the previous observations based on the defined schedule, aggregates them, clears the temporary observation table, and publishes the aggregated payload back to the message broker. Another subscriber is the speed estimation engine which handles vehicle line crossing, coordinates projection, trajectory refinement, and speed estimation. This engine updates the temporary observation table and removes observations right after finishing the speed estimation. Both engines publish traffic parameters to the message broker, and the permanent observation table will be updated by the camera id primary key and the capture time sort key. This makes the observations easy to query where users can receive traffic values for each camera at a specific time.

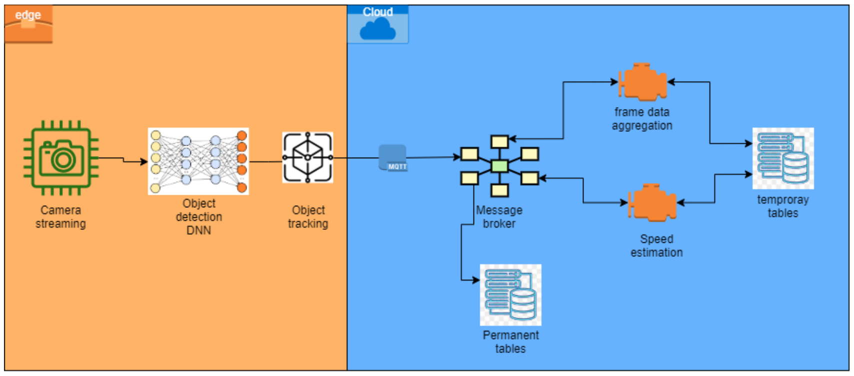

Figure3- Proposed traffic estimation architecture

2.3. Data and implementations

To implement the proposed architecture, we used a USB camera to capture the online street feeds. As the edge computer, we used Jetson Nano with NVIDIA Maxwell architecture with 128 NVIDIA CUDA® cores GPU, Quad-core ARM Cortex-A57 MPCore processor CPU, 4 GB RAM, and two frames per second processing speed for object detection. The DNN architecture was Yolov3 and Amazon Web Service (AWS) IoT core was used as the message broker. AWS Lambda functions were employed to run the processes and AWS DynamoDB was storing the temporary and permanent observations. The data used in this paper is provided by the OGC Smart City Interoperability Reference Architecture (SCIRA) project working group from Calgary streets camera feeds (Saeedi et al., 2020).

3. Results

Table 1 provides the details of the proposed architecture results. Precision is used as the evaluation metric to demonstrate the accuracy of the traffic parameters. The field observations are used to measure True positives and False positives. For the performance, the time that the entire traffic estimation process took (i.e., from the object detection step to traffic parameter storing) is considered.

Table 1: The proposed architecture results

Item	Value
Cars number (precision)	93%
Pedestrian number (precision)	87%
Car speed estimation (precision)	78%
Time Performance	846 milliseconds

4. Discussion & Conclusion

This paper proposed an architecture to estimate traffic in real-time with high accuracy. The architecture is implemented on the Nvidia Jetson Nano edge device, which receives camera feeds as input, detects objects, identifies them, then aggregates them to estimate the number of cars and pedestrians in a specific street. The proposed methodology estimates cars' speed by projecting the car's trajectories into geographic space and tracking them until they pass a predefined region in the frames. Results demonstrate a higher accuracy for car counting rather than other traffic parameters (i.e., pedestrian counting and speed estimation). It is because of the higher accuracy of the DNN model as well as the simpler aggregation method rather than other parameters. Speed estimation has the least precision since it involves more complicated calculations such as trajectory refinement, coordinates projection, and length measurements. The architecture returns traffic estimation results in less than a second which means the capability of estimating the traffic parameters in near real-time. The proposed method provides higher performance for the real-time traffic estimation in comparison with the traditional cloud computing or edge computing methods. The proposed solution could be used in Traffic estimation and prediction systems (TrEPS), real-time traffic monitoring, and traffic control in urban areas.

References

Barrios, C., & Motai, Y. (2011). Improving estimation of vehicle's trajectory using the latest global positioning system with Kalman filtering. *IEEE Transactions on Instrumentation and Measurement*, 60(12), 3747-3755.

Barthélemy, J., Verstaevel, N., Forehead, H., & Perez, P. (2019). Edge-computing video analytics for real-time traffic monitoring in a smart city. *Sensors*, 19(9), 2048.

Bewley, A., Ge, Z., Ott, L., Ramos, F., & Upcroft, B. (2016, September). Simple online and realtime tracking. *In 2016 IEEE international conference on image processing (ICIP)* (pp. 3464-3468). IEEE.

Honarparvar, S., Saeedi, S., Liang, S., & Squires, J. (2021). Design and Development of an Internet of Smart Cameras Solution for Complex Event Detection in COVID-19 Risk Behaviour Recognition. *ISPRS International Journal of Geo-Information*, 10(2), 81.

Kar, G., Jain, S., Gruteser, M., Bai, F., & Govindan, R. (2017, October). Real-time traffic estimation at vehicular edge nodes. *In Proceedings of the Second ACM/IEEE Symposium on Edge Computing* (pp. 1-13).

Onica, E., Felber, P., Mercier, H., & Rivière, E. (2016). Confidentiality-preserving publish/subscribe: A survey. *ACM computing surveys (CSUR)*, 49(2), 1-43.

Papageorgiou, M. (Ed.). (1983). Applications of automatic control concepts to traffic flow modeling and control. Berlin, Heidelberg: Springer Berlin Heidelberg.

Papageorgiou, M., Diakaki, C., Dinopoulou, V., Kotsialos, A., & Wang, Y. (2003). Review of road traffic control strategies. *Proceedings of the IEEE*, 91(12), 2043-2067.

Redmon, J., & Farhadi, A. (2018). Yolov3: An incremental improvement. arXiv preprint arXiv:1804.02767.

S. Saeedi, B. Lieberman, J., Liang, S., Hawkins, C. Chen, I. Correas, I. Starkov, J. MacDonald, M. Alzona, M. Botts, M. Jahromi, S. Honarparvar, Maddala S (2020). OGC Smart City Interoperability Reference Architecture (SCIRA) Pilot Engineering Report. *OGC*