2016 IEEE Second International Conference on Multimedia Big Data

Dynamic Urban Surveillance Video Stream Processing Using Fog Computing

Ning Chen[†], Yu Chen[†], Yang You[†], Haibin Ling[ξ], Pengpeng Liang[ξ], Roger Zimmermann[‡]
[†]Dept. of Electrical and Computing Engineering, Binghamton University, Binghamton, NY. USA 13902
[ξ]Dept. of Computer and Information Sciences, Temple University, Philadelphia, PA. USA 19122
[‡]Department of Computer Science, National University of Singapore, Singapore 117417

***Abstract*—The recent rapid development of urbanization and Internet of things (IoT) encourages more and more research on Smart City in which computing devices are widely distributed and huge amount of dynamic real-time data are collected and processed. Although vast volume of dynamic data are available for extracting new living patterns and making urban plans, efficient data processing and instant decision making are still key issues, especially in emergency situations requesting quick response with low latency. Fog Computing, as the extension of Cloud Computing, enables the computing tasks accomplished directly at the edge of the network and is characterized as low latency and real time computing. However, it is non-trivial to coordinate highly heterogeneous Fog Computing nodes to function as a homogeneous platform. In this paper, taking urban traffic surveillance as a case study, a dynamic video stream processing scheme is proposed to meet the requirements of real-time information processing and decision making. Furthermore, we have explored the potential to enable multi-target tracking function using a simpler single target tracking algorithm. A prototype is built and the performance is evaluated. The experimental results show that our scheme is a promising solution for smart urban surveillance applications.**

***Index Terms*—Smart City, Urban Surveillance, Fog Computing, real-time processing, Speeding Traffic.**

I. Introduction

With the increasing urbanization and prosperity of the Internet of things (IoT), cities are making their way to be even much smarter [3], [4], [36]. It is predicted that by 2020 there will be more than 10 billion mobile devices that produce tons of data every day and trillions of sensors that will monitor and communicate with each other, flooding the IoT with dynamic real-time data [16]. These ubiquitously distributed sensors and smart devices bring smart cities a broad variety of data from which urban planners can obtain timely living patterns updating [1]. Urban surveillance, as an essential part of situational awareness for better urban management and planning, deals with heterogeneous data from a layered sensors environment [5]. For object assessment and target tracking, information fusion is indispensable. Efficient extracting, analyzing and understanding the large scale data set from heterogeneous smart devices in a real-time manner are essential in mission critical applications, such as the instant decision making in emergency situations. However, there is still a huge performance gap between the amount of data and the lack of adequate resources at the edge of network [8].

For urban surveillance tasks requiring complex data fusion, Cloud Computing has been widely recognized as the solution. However, Cloud Computing is not the silver bullet that works for all kind of applications. The extra round-trip delays and possible network congestions are not tolerable in some latency sensitive applications, such as real-time raw video streaming mining. Fog Computing [6], [35], one extension of Cloud Computing paradigm, is a promising solution to the mission critical tasks involving information fusion, quick decision making and situation awareness. Instead of transmitting collected data to remote Cloud center, Fog Computing leverages computing resources at the edge of the network, i.e. the embedded and mobile computing devices carried by end users.

Urban traffic surveillance, with the help of massive trajectory data collected from pervasively deployed sensors, is of great value. It enables city administration and law enforcement department get information quickly and allocate the resources efficiently. For example, over-speed driving violation brings unpredictable danger to the drivers and could be fatal to innocent people such as the pedestrians. Therefore, a smart, real-time speeding traffic monitoring system would be very helpful to reduce the number of car accidents.

In this paper, we propose an urban speeding traffic monitoring system using Fog Computing paradigm. A drone is used to monitor the vehicles on the roads and the video is sent back to the controller on the ground. Due to the limited computing capability of the controller, the raw video stream was sent to a Fog Computing node, where the moving vehicle tracking algorithms are executed. Since we have already verified the correctness of the proposed system [8], in this work, the focus is to verify that the performance of our monitoring system meets the requirement of real-time monitoring and instant decision making. Considering the size of the video frames, an efficiently dynamic real-time frame processing scheme was proposed. Leveraging the divide-and-conquer strategy, the subarea containing the vehicle of interests was identified and transmitted to the Fog node for processing. After the processing of the subarea in computing units, the tracking result would be sent back to end users and will be synchronized with the remaining part of that frame to display. The experimental results are very encouraging.

The rest of this paper is organized as follows. Section 2 provides a brief discussion of closely related work about Fog Computing and traffic monitoring. Section 3 introduces the Fog Computing based real-time speeding traffic surveillance system. Section 4 shortly describes the tracking algorithm. Section 5 reports the detailed experimental results. Section 6 concludes this paper with some discussions.

978-1-5090-2179-6/16 $31.00 © 2016 IEEE
DOI 10.1109/BigMM.2016.53

II. Related Work

Because of advantages such as flexibility, safety and easy to manipulate, quadcopter drones have been widely utilized to assist people in multiple areas, including service improvement, urban surveillance and scientific research. Particularly, drones are ideal tool for dull, dirty or dangerous work that may cause harmful consequences to human operators. Recently, the opportunities for UAVs to be used in smart cities are discussed [24]. Researchers have recognized that UAVs could be utilized in a wide range of urban applications such as geo-spatial and surveying activities, traffic and crowd management, natural disaster control and monitoring and Big Data processing. Combining the wireless networks or mobile applications, UAVs help the police department maintain the safety and security in the urban residence. In 2010, UK policemen arrested a car thief suspect with the help of a UAV [11].

In addition, the UAVs can be used for remote sensing and photogrammetry [10], [13], [17], [18]. Niethammer and colleagues [25] use UAVs to map landslides with high resolution, the results are encouraging but the improvements are still needed to reduce the image processing time. These applications indicate that UAVs, as the sensors in the sky, can provide valuable data for urban surveillance tasks and could alleviate the problem of data sparsity. However, lack of real-time processing capability is still an obstacle to make full use of the abundant data collected by UAVs.

One example application of UAV data is moving target monitoring and activity recognition using wide area motion imagery (WAMI) [5], [9], [21], [26], [28], [31], [33]. WAMI is characterized by its high data rate and the wide area coverage, which provides plentiful information. Real-time information fusion is of great importance for the situational awareness in urban surveillance tasks [20]. However, due to the increasingly big size of real-time video and imagery data, it is very challenging to achieve the goal of real-time information fusion. A container-based cloud architecture has been proposed for pseudo real-time target tracking in full-motion video and WAMI stream [32]. Besides the Cloud Computing platform proposed in [32], the performance of visual tracking in extremely low frame rate WAMI was evaluated in [19]. Though a variety of methods have been proposed to reduce the processing time, the major concern about the Cloud based solution lies in the high latency introduced by data transmission to and from far away Cloud Computing center. In addition, low frame rate would reduce the valuable information for situation awareness and decision making. Therefore, an improvement is necessary to achieve the goal of real-time processing in urban surveillance tasks.

Fog computing recently appeared promising to meet the requirement of real-time data fusion for dynamic urban surveillance. The IoT and ubiquitous sensors have pushed the computing to the edge of network. The most explicit difference between Fog Computing and Cloud Computing is that Fog Computing provides computing facility nearby that enables on-site real-time analysis and instant decision making. Researchers have explored application of Fog Computing [7], [12], [27], [34]. Fog Computing has been applied in IoT for healthcare, ECG feature extraction is taken as a case study to verify the feasibility [14]. In [29], a hierarchical distributed Fog Computing architecture is studied for Big Data analysis in smart cities. In [15], [22], researchers explore how the Fog Computing can efficiently reduce the mobile data traffic and enhance the quality of service for mobile users.

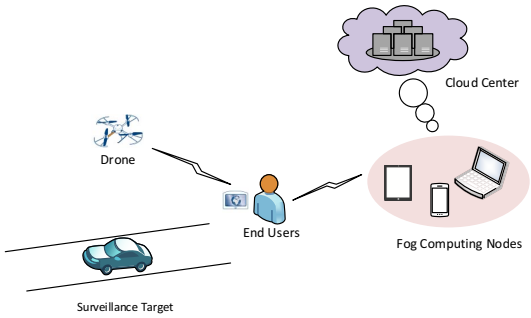

Fig. 1. System Architecture.

III. System Architecture

Figure 1 illustrates the proposed three-layer urban surveillance system architecture, which consists of surveillance application layer (also be called user layer), Fog Computing layer, and Cloud Computing layer. The on-site or near-site Fog Computing layer is of the greatest importance for real-time data processing. A wide range of smart devices serve as Fog Computing nodes, i.e. smart tablets, personal smart phones, the laptop in the police car, or on-board computing device on the drone. When the raw video streams are collected, instead of transferring them to the remote Cloud center, the processing tasks are assigned to near-site Fog computing devices. Thus, the latency of transmitting data from surveillance area to the Cloud center is removed. Also, the Fog Computing layer prevents the local significant data from being sent to the Cloud. It reduces the work load of the communication network.

The video processing time at the Fog nodes is another key issue for the real-time processing. In order to meet the real-time video stream processing requirement, the output frame rate should be equal to or higher than the input frame rate. To achieve this goal, two methods are often utilized by researchers. One is to decrease the video frame rate or discard some frames. Although the performance of this strategy is acceptable, the big gap between two consecutive frames would cause the loss of suspicious targets since the target may move large distance between these two subsequent frames and some threats can be hidden. Another way is to decrease the resolution of surveillance video. Lower video resolution does reduce the data size, but it sacrifices the details in video stream which can be a big loss of information, especially in some security or safety related applications. The higher resolution that a surveillance video has, the more information for the situational awareness and decision making.

In the proposed surveillance system architecture, a drone acts as a sensor to monitor the area of interests. Once the

surveillance video data are generated, the raw stream is sent back to the ground controller station and display on a screen. The operator, i.e. a police officer, once find a suspicious vehicle driving very fast, he can lock that vehicle in the real-time video for further tracking. The tracking algorithm is executed at the near-site Fog Computing nodes in which each of the video consecutive frames are processed. In our dynamic stream processing scheme, instead of forwarding the whole video frame, the sub-area including the suspicious vehicle is extracted from the original frame and sent to the Fog Computing units. The size of the sub-area of interest is determined by the computing resource provided by the Fog Computing nodes. Our scheme reduces the processing time at the computing nodes, and also cuts down the data size to be transmitted which can reduce the transmission time.

When the speed is calculated, the result and the processed sub-area will be sent back to the ground controller station immediately. The sub-area would be synchronized with remaining part of the frame and displayed on the screen. The Fog Computing nodes not only provides the computing resource, but also the storage space. In this urban speeding surveillance system, the processed vehicle motion data can be saved in Fog nodes for a short period and then would be uploaded to the Cloud center for a long-term analysis. For example, in our case, research about during what time the over-speed driving could happen most likely can be of interest.

IV. Tracking Algorithms

In practical scenarios, when a suspicious vehicle is considered in the real-time surveillance video, it needs to be locked immediately and tracked with high accuracy frame by frame. The vehicles are not the only things that appear in the surveillance video, there are also occlusion, background clutter, the variation of illumination and even noise in the practical scenarios, which would affect the tracking accuracy and efficiency. A robust and effect tracking algorithm is highly desired based on which the precise speed information can be calculated.

In our speeding vehicle surveillance system, based on the specific requirements in the practical scenarios, a robust L1 tracker using accelerated proximal gradient approach is adopted [2]. This algorithm is casted by the sparse representation in the particle filter framework.

A. Particle Filter

The particle filter implements the recursive Bayes estimation using the method of non-parametric Monte Carlo simulation. It uses a large number of particles that are transferred in the state space to estimate the probability density function of state variables. Particle filter is an efficient tool to solve the problem in non-linear system. In addition, the distribution of random variables are unnecessary to be Gaussian distribution. Two steps are essentially involved in the particle filter: prediction and update.

We denote x_t to represent the state variable describing the motion of the target in frame t. y_t denotes the observation of the moving target in frame t. In target tracking applications, we assume state variable x_t is only related to x_{t-1} and the observation at frame t is only related to x_t, which means observations among $y_{1:t} = \{y_1, y_2, \cdots, y_t\}$ are independent of each other. It is assumed that at frame $t-1$, the probability density distribution is $p(x_{t-1}|y_{t-1})$. In prediction step, $p(x_t|y_{t-1})$ is derived from $p(x_{t-1}|y_{t-1})$:

$$p(x_t, x_{t-1}|y_{t-1}) = p(x_t|x_{t-1}, y_{t-1})p(x_{t-1}|y_{t-1}) \quad (1)$$

Given x_{t-1}, x_t and y_{t-1} are independent, Eq. (1) becomes:

$$p(x_t, x_{t-1}|y_{t-1}) = p(x_t|x_{t-1})p(x_{t-1}|y_{t-1}) \quad (2)$$

Then compute the integration of Eq. (2) over x_{t-1}:

$$p(x_t|y_{t_1}) = \int p(x_t|x_{t-1})p(x_{t-1}|y_{t-1})dx_{t-1} \quad (3)$$

With Eq. (3), we can move forward to the update step by using Bayes rules:

$$p(x_t|z_t) = \frac{p(x_t|x_{t-1})p(x_t|y_{t-1})}{p(y_t|y_{t-1})} \quad (4)$$

where $p(y_t|x_t)$ is the observation likelihood. In the particle filter, the posterior probability above is estimated by N samples, denoted by $S_t = \{x_t^1, x_t^2, x_t^3, \cdots, x_t^N\}$ with different weights. Due to the lack of knowledge about variable distribution, sequential important distribution $q(x_t^{(i)}|y_t)$ is used to generate the samples. The weight is:

$$W_t^{(i)} \propto \frac{p(x_t^{(i)}|y_t)}{q(x_t^{(i)}|y_t)} \quad (5)$$

and the weight can be updated as follows:

$$W_t^i = w_{t-1}^i \frac{p(y_t|x_t^i)p(x_t^i|x_{t-1}^i)}{q(x_t|x_{1:t-1}, y_{1:t})} \quad (6)$$

The observation likelihood depicts the similarity between the target candidate and the target templates [23].

B. Modified L1 Minimization Tracker

In sparse approximation, the signal y can be linearly represented by the atoms of the over-complete dictionary D.

$$y = D \cdot x \quad (7)$$

where x is the coeffieient of each atom in the dictionary D. In moving target tracking algorithm, over-complete dictionary consists of target templates denoted by $T = t_1, t_2, t_3, \cdots, t_n$. With the target templates, a target candidate can be represented as follows:

$$y \approx T \cdot x = x_1 t_1 + x_2 t_2 + \cdots + x_n t_n \quad (8)$$

Because of the sparsity in sparse approximation, for a good target candidate, most coefficients of the target templates should be zero, which means a good target candidate can be nearly represented by several target templates. In other words, the coefficients of a bad target candidate can be relatively of smaller number.

In the real scenarios of our monitoring videos, we have to consider the errors resulted from objects other than the target, such as occlusion, noise, shadows, sometimes even darkness. Therefore, trivial templates denoted by $I = i_1, i_2, i_3, \cdots, i_n$

are introduced in this algorithm and the Eq. (8) is rewritten as follow:

$$y = \begin{pmatrix} T & I \end{pmatrix} \begin{pmatrix} x \\ e \end{pmatrix} \tag{9}$$

where e represents the coefficients of trivial templates. In a further consideration, it is reasonable to assume that the coefficients of a good candidate should be positive, which can also be considered as the non-negative constraints. Hence, in this scenario, positive and negative trivial templates should be involved. Then Eq. (9) is rewritten as:

$$y = \begin{pmatrix} T & I & -I \end{pmatrix} \begin{pmatrix} x \\ e_+ \\ e_- \end{pmatrix} = D \cdot m \tag{10}$$

Here, $D = (T\ I\ -I)$ and $m^T = (x\ e_+\ e_-)$. What we want to know is the coefficients m of the target templates and trivial templates, but in the over-complete dictionary $D^{m \times n}$, m is much smaller than n, which means the solution of Eq. (10) is not unique. Some constraints are indispensable in order to get a unique solution in the sparse representation. Fortunately, we can solve this problem as an $L1$ norm least squares problem.

$$min \|Dm - y\|_2^2 + \lambda \|m\|_1 \tag{11}$$

where $\|\cdot\|_1$ denotes l_1 norm and $\|\cdot\|_2$ denotes l_2 norm respectively. As mentioned above, trivial templates are brought into the dictionary to deal with the noise and occlusion. But what if there is no occlusion? The target object should be well approximated by the target templates from previous frames. In case of no occlusion in the frame, the trivial templates would impact the detection accuracy otherwise and bring some computation complexity. So in this accelerated $l_!$ norm tracking algorithm, another coefficient μ_t is introduced to improve the constraint (11). The revised constraint is as:

$$min \frac{1}{2} \|y_t - Dm\|_2^2 + \lambda \|m\|_1 + \frac{\mu_t}{2} \|m_I\|_2^2 \tag{12}$$

where m_I is the coefficients of trivial templates in this target tracking sparse approximation problem: $m = [m_T\ m_I]$. If occlusion is detected in a video frame, μ_t is zero. Otherwise, μ_t is supposed to be certain specific value.

In practical experiments, solving such kind of modified l_1 norm minimization could be pretty time consuming. A fast numerical method called accelerated proximal gradient [30] is applied to solve this problem. This approach is designed for solving the optimization problem as below:

$$min\ F(a) + G(a) \tag{13}$$

and the accented proximal gradient is fast for some specific types of function G.

After solving the l_1 least squares minimization problem and obtaining the sparse coefficients m, the observation likelihood of state variable x_t^i can be expressed as:

$$p(y_t | x_t^i) = \frac{1}{\Gamma} exp\{-\alpha \|y_t^i - T_t m_T^i\|_2^2\} \tag{14}$$

where α is used to control the shape of a Gaussian Distribution and Γ is a normalized factor. m_T denoted the coefficients of target templates. The optimal state x_t^i satisfies:

$$x_t^i = \underset{x_t^i \in S_t}{\arg\max}\ p(y_t | x_t^i) \tag{15}$$

Fig. 2. A surveillance video frame

V. Experimental Results

We have tested the performance of our proposed Fog Computing based urban speeding surveillance system and the dynamic surveillance video processing scheme. The experimental results are reported in this section.

A. Experimental Setup

A prototype of our proposed system has been built. Two DJI Phantom 3 Professional drones are used to monitor the moving vehicles on road and two Nexus 9 tablets are connected to the drone controllers to display the real-time surveillance video. In our prototype, one laptop acts as a Fog Computing node whose configuration is as follows: the processor is 2.3 GHz Intel Core i7, the RAM memory is 16 GB and the operating system is OS X EI Capitan. The resolution of our video frames is 1280×720. The OpenCV 3.1 and Eigen 3.2.1 are used for the tracking algorithm. The given FOV (field of view) of the camera mounted on the drones is $94°$, and the actual FOV after calibration is $89.39°$ according to the fact that manufacturers would always make the image plane not perfectly circumscribed with the CCD plate but a little larger than that.

There are two video streams used. One is taken by the drone above an on-campus road of Binghamton university, where the speed limit is 30 mile per hour. A black Toyota Camry is moving on the road with the constant speed of 27 mile per hour for the speed calculation accuracy test. There are 514 frames in this video stream. Another video stream is obtained above the I-81 highway from Binghamton to Syracuse, which is used to evaluate the performance of our system. We would use the gray information of the video frames to track the vehicles and the video frames are stored in JPG format. Figure 2 is an example frame, the car in the white bounding box is that Toyota Camry with the constant speed 27 miles per hour.

B. Performance Evaluation

It is a challenge for our tracking function to work in a noisy environment, in which there are multiple similar vehicles as the tracking vehicle, the occlusion, the shadows, etc. It is critical to ensure that the tracking algorithm can properly and

Fig. 3. Tracking Test Sequence 1.

robustly track the suspicious target and the speed is calculated correctly. An accurate speed assessment is equally important as the police officer will make decisions on the further actions if necessary.

Figure 3 shows the tracking results of a moving target on the road with shadow of trees along the roadside. The black Toyota Camry is the target and a red bounding box is on its body in the image showing it is locked. The vertical height from on-board camera to the ground is 140.0 meters. The tracking results have shown that our scheme can lock the target all the time without losing the track. The vehicle moving speed is calculated based on the position of the target in each frame. Knowing the height and the FOV of the camera, the diameter of the object circle plane can be calculated, which is also the real distance the diagonal of the image represents. Further, we can obtain the unit distance that one pixel represents. Combining the unit distance and the interval time slot, the speed can be easily obtained. More details and the explicit algorithm explanation can be found in our previous work [8].

Figure 4 presents the estimated speed results. The video sequence contains 511 frames and the speed of the tracked vehicle was calculated every fifteen frames, which means the time resolution is 0.5 second. As shown in Fig. 4, the estimated results stay close to the actual speed, which is 27 mile per hour and the results varies in the range from 25 to 30 mph. The largest speed is 28.97 mph and the lowest speed is 24.97 mph.

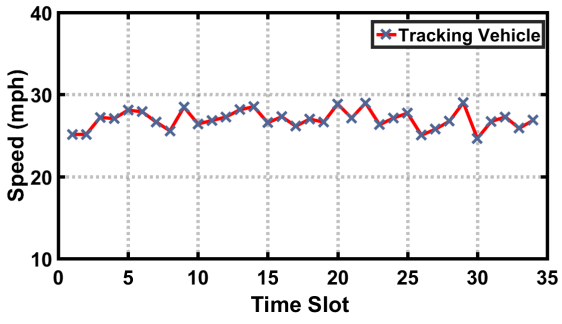

Fig. 4. Speed calculation results.

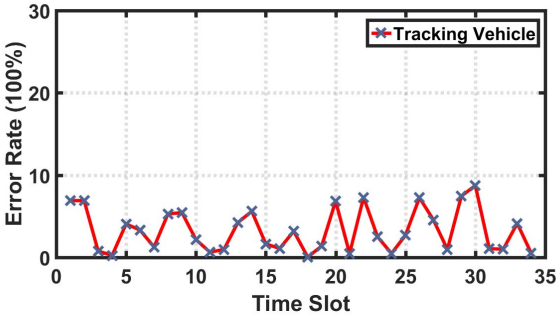

Fig. 5. Error rate.

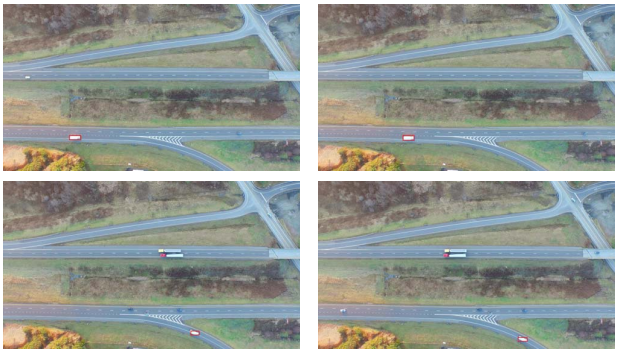

Fig. 6. Tracking Test Sequence 2 on highway.

The error is define as follow:

$$error = \frac{|estimation - actual|}{actual} \tag{16}$$

Figure 5 provides a better view of the results. The detection error compared to the real speed of the vehicle stays below 10% all the time. The Figs. 4 and 5 demonstrate that the proposed detection system can efficiently track the vehicle and obtain the speed with the acceptable accuracy.

Another tracking experiment has been conducted using a video stream monitoring freeway I-81 from Binghamton to Syracuse, the speed limit is 65 mph. The target is a white truck what is entering a ramp exiting the highway. The tracking results are shown in Fig. 6 and the speed calculation results are shown in Fig. 7. Figure 6 verified that our tracking algorithm can properly track a vehicle that is changing its direction. In this scenario, the locked vehicle is going to leave the highway. Based on our driving experience, the speed should be decreased during a turning. As shown in Fig. 7, at the first time point, the speed is close to 62 mph, indicating that the vehicle started to slow down. Then the red line keeps going down along with the time depicting that the white truck is leaving the highway with slower and slower speed.

C. Dynamic Frame Processing

The above study has demonstrated that our proposed speeding vehicle detection system can successfully identify suspicious target and track it correctly. The next critical question we

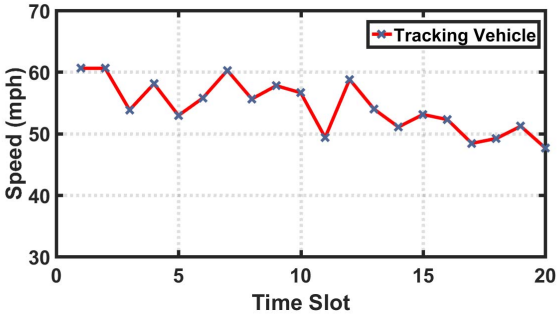

Fig. 7. Speed estimation on highway.

Fig. 8. Example of definition of sub-areas.

need to answer is whether or not our system is able to process the surveillance video stream fast enough to accommodate the performance requirement for real-time surveillance.

It took 29.35 seconds to process the first test video stream, which resolution is 1280×720 and there are 511 frames in total. The average processing time for each frame is 57.5 ms, which means that the system can process 17.4 frames per second. Regarding the second video stream, it took 19.47 seconds for 240 frames, the resolution is 4096×2160. The average processing time is 81.1 ms. The frame rate of the video is 30 frames per second. Therefore, it implies an optimized approach is needed to achieve the goal of real-time processing.

Intuitively, the barriers that prevent the system from achieving real-time processing mainly come from two parts: the processing time at the Fog Computing nodes and the transmission time of the surveillance data from the collecting device to the computing units. The larger size the data is, the longer is transmission delay. It is worthy to note that in the era of IoT, hundreds of thousands devices are connected together, with huge amount of data produced every minute. Such high data volume can easily cause network congestion without a well designed traffic manage scheme.

In order to address these challenges, a dynamic frame processing scheme is proposed. It not only aims at reducing the computing time at each individual Fog node, but also transmits smaller amount of data, which in turn would decrease the transmission time and alleviate the network work load. To achieve these goals, a divide-and-conquer strategy is adopted in our dynamic processing scheme. Instead of assigning a full video frame to a Fog node at one time, a sub-area containing the vehicle of interests is identified and transmitted to a computing node to be processed.

Figure 8 shows an example of making sub-areas with different colors and different sizes. Theoretically speaking, the smaller size of the data to be processed means less time to accomplish the tracking task. For the sake of convenience, the selected sub-area is specified as a square calculated using the size of the vehicle as the basic unit. For example, assume the length of the monitored vehicle is one, then the size of the square area can be $4(2 \times 2)$, $9(3 \times 3)$, $16(4 \times 4)$, etc.

Figure 9 illustrates the results of dynamic frame processing scheme using the test sequence 1. The X axis represents the size of the sub-area in the bounding box used for locking the

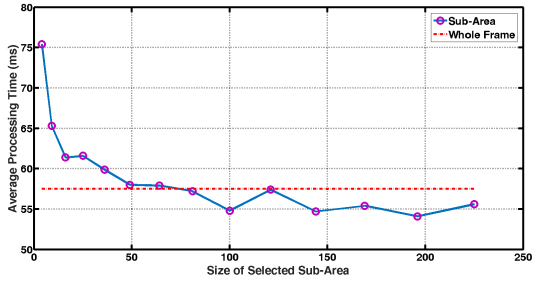

Fig. 9. Differing sub-area test sequence 1.

suspicious speeding vehicle. The Y axis represents the average processing time for each frame in the video sequence. The red line marks the baseline representing the average processing time without dynamic sub-area policy applied and the blue line depicts the average computing time with different sizes of sub-areas. The size one is not considered since that sub-area is too small to be processed.

As shown in Fig. 9, it is interesting that the smallest sub-area strategy takes the longest processing time. It is counter-intuitive and different from our initial conceiving. The average processing time goes down as the sub-area increases until the size becomes 49 (7×7), then the average time stays around the average processing time achieved without the sub-area policy applied.

Figure 10 presents the results using the test video sequence 2, in which the axis is the same as Fig. 9. Similarly, the smallest sub-area is still characterized as the time consuming in the whole figure. But when the size of the bounding box area becomes 49 (7×7), the average processing time is close to 70.5 millisecond for each frame and the frame rate is 14.18. Comparing with the average time of whole frame processing, the sub-area with size 7×7 can effectively reduce the processing time, around 13% down. As the size continues going up, the average processing time increases again and eventually would larger than the time without sub-area policy.

With a deeper analysis of the behavior of our tracking algorithm, this phenomenon is resulted from two factors. The first one is the overhead incurred by re-initialization of the tracker every time a new sub-area is assigned. The tracker needs to learn the background and the characteristics

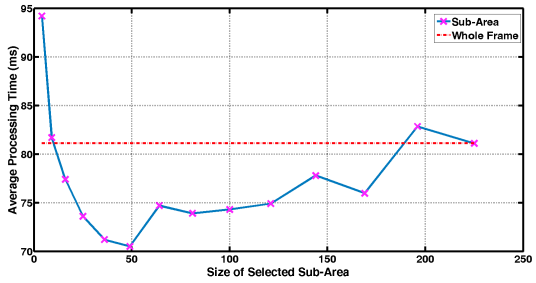

Fig. 10. Differing sub-area test sequence 2.

of the target. While the smaller sub-area shrinks the data size for transmitting and processing, the tradeoff is that each frame is divided into more sub-areas. Every time, when the tracker deals with a new sub-area, it will discard the previous templates and treats the new sub-area as a new one. This operation is computationally expensive with more resource consumed. Hence, the processing time is actually longer when smaller sub-areas are adopted. The sub-area size of 7×7 is close to the actual area in the frame utilized by the tracker.

The other factor is that the tracking problem is handled as a sparse representation within particle filter framework. The tracker updates the templates along with the tracking process for higher accuracy. Particularly, the particles follow Gaussian distribution. Thus, large amount of particles stay closely around the circle center point to detect the similarity between tracking candidates and the original tracking target. The circle center point in our tracking process is the center point of our bounding box and there would be no particles far away from the center point with low probability. Therefore, the tracking algorithm already handles the frame in a way that is similar to the dynamic sub-area based tracking.

The resolution of test sequence 2 is 4094×2160, which is larger than the resolution of sequence 1. Before the sub-area size becomes 7×7, the trend of average processing time stays as same as that in Fig. 9. Difference appears when the sub-area size is larger than 7×7. The average time starts to grow from 70.5 ms, approaching to 81.1 ms, which is the average processing time of an entire frame.

The convex-like curve in Fig. 10 appears mainly because of the higher resolution. The sub-area size of 7×7 in this scenario is smaller than the area the tracker used in tracking. As the size of sub-area increases, the processing time becomes longer as well. Considering the effects shown in the two figures, it is clear that the smaller sub-area does reduce the data size of each individual job, but could also increase the total tracking time. The data size and the processing time need to be balanced to find an optimal operation point.

D. Multiple Vehicles Tracking

In practical scenarios, generally there are more than one vehicles need to be tracked simultaneously. While the multiple target tracking is necessary, it is still an unmatured research area. The dynamic sub-area assignment mechanism introduced in our proposed Fog Computing based surveillance system

Fig. 11. Two vehicles tracking.

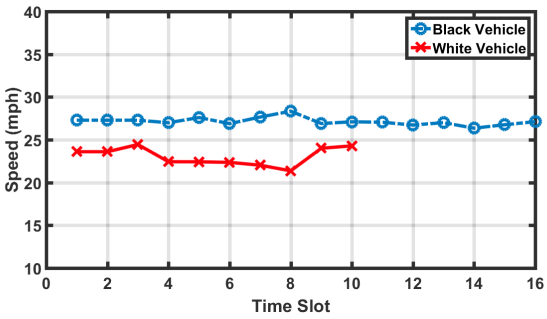

Fig. 12. Multiple vehicles detection results.

is able to track multiple vehicles in parallel, although it is a single target tracking algorithm adopted in our system. When there are two or more vehicles need to be tracked, same processing scheme is applied to each individual sub-area containing the vehicle of interests. Each sub-area image will be sent to different Fog Computing nodes and they are processed in parallel. A preliminary experimental study has been conducted to verify the feasibility.

Figure 11 shows a scenario in which two vehicles are tracked and the testing results are shown in Fig. 12. Red line represents the white vehicle and the blue line represents the black vehicle. Each time slot is one second. As shown by Fig. 12, the speed of the black vehicle stays around 27 mile per hour and the white vehicle speed stays around 23 mile per hour. The red line ends at 10 second point because the white vehicle went out of the image and the tracker stopped tracking. This simple experimental study verified that the proposed system is able to track multiple vehicles utilizing the advantages of Fog Computing.

VI. Conclusion

In this paper, we propose a smart city speeding traffic surveillance scheme using Fog Computing paradigm. A prototype has been built in which two DJI drones are integrated for monitoring and one laptop serves as a Fog Computing node. Intensive experiments are conducted using real-world traffic surveillance video streams. The experimental results have validated the effectiveness of our system. A dynamic

sub-area of interest assignment scheme is suggested to promote the performance to meet the requirements of real-time surveillance tasks. A balance between the sub-area size and the processing time is discussed based on the numerical testing results. Furthermore, we have explored the feasibility of concurrent multiple targets tracking using the single target tracking algorithm leveraging the divide-and-conquer strategy in the Fog. The result is very encouraging, showing that our system has the potential to handle multiple targets without using more complex multi-target tracking algorithm. The ongoing efforts focus on two important issues: (i) reducing the overhead incurred by tracker initialization when a new sub-area of interest is assigned; and (ii) implementing and evaluating the multi-target tracking scheme using concurrent multiple single target tracking jobs.

REFERENCES

[1] D. Arribas-Bel, "Accidental, open and everywhere: Emerging data sources for the understanding of cities," *Applied Geography*, vol. 49, pp. 45–53, 2014.

[2] C. Bao, Y. Wu, H. Ling, and H. Ji, "Real time robust l1 tracker using accelerated proximal gradient approach," in *Computer Vision and Pattern Recognition (CVPR), 2012 IEEE Conference on*. IEEE, 2012, pp. 1830–1837.

[3] M. Batty, "The pulse of the city," *Environment and Planning B: Planning and Design*, vol. 37, no. 4, pp. 575–577, 2010.

[4] M. Batty *et al.*, "Smart cities, big data," *Environment and Planning-Part B*, vol. 39, no. 2, p. 191, 2012.

[5] E. Blasch, G. Seetharaman, S. Suddarth, K. Palaniappan, G. Chen, H. Ling, and A. Basharat, "Summary of methods in wide-area motion imagery (wami)," in *SPIE Defense+ Security*. International Society for Optics and Photonics, 2014, pp. 90 890C–90 890C.

[6] F. Bonomi, R. Milito, J. Zhu, and S. Addepalli, "Fog computing and its role in the internet of things," in *Proceedings of the first edition of the MCC workshop on Mobile cloud computing*. ACM, 2012, pp. 13–16.

[7] Y. Cao, S. Chen, P. Hou, and D. Brown, "Fast: A fog computing assisted distributed analytics system to monitor fall for stroke mitigation," in *Networking, Architecture and Storage (NAS), 2015 IEEE International Conference on*. IEEE, 2015, pp. 2–11.

[8] N. Chen, Y. Chen, X. Ye, H. Ling, P. Liang, S. Song, and C.-T. Huang, "Smart urban surveillance using fog computing: a case study on speeding traffic detection," in *Technical Report, Department of Electrical and Computer Engineering, Binghamton University*, 2016.

[9] J. Choi, Y. Dumortier, J. Prokaj, and G. Medioni, "Activity recognition in wide aerial video surveillance using entity relationship models," in *Proceedings of the 20th International Conference on Advances in Geographic Information Systems*. ACM, 2012, pp. 466–469.

[10] I. Colomina and P. Molina, "Unmanned aerial systems for photogrammetry and remote sensing: A review," *ISPRS Journal of Photogrammetry and Remote Sensing*, vol. 92, pp. 79–97, 2014.

[11] DailyMail.com, "Drone makes first uk 'arrest' as police catch car thief hiding under bushes http://www.dailymail.co.uk/news/article-1250177/Police-make-arrest-using-unmanned-drone.html," access February 11, 2016.

[12] S. K. Datta, C. Bonnet, and J. Haerri, "Fog computing architecture to enable consumer centric internet of things services," in *Consumer Electronics (ISCE), 2015 IEEE International Symposium on*. IEEE, 2015, pp. 1–2.

[13] J. Everaerts *et al.*, "The use of unmanned aerial vehicles (uavs) for remote sensing and mapping," *The International Archives of the Photogrammetry, Remote Sensing and Spatial Information Sciences*, vol. 37, pp. 1187–1192, 2008.

[14] T. N. Gia, M. Jiang, A.-M. Rahmani, T. Westerlund, P. Liljeberg, and H. Tenhunen, "Fog computing in healthcare internet of things: A case study on ecg feature extraction," in *Computer and Information Technology; Ubiquitous Computing and Communications; Dependable, Autonomic and Secure Computing; Pervasive Intelligence and Computing (CIT/IUCC/DASC/PICOM), 2015 IEEE International Conference on*. IEEE, 2015, pp. 356–363.

[15] M. A. Hassan, M. Xiao, Q. Wei, and S. Chen, "Help your mobile applications with fog computing," in *Sensing, Communication, and Networking-Workshops (SECON Workshops), 2015 12th Annual IEEE International Conference on*. IEEE, 2015, pp. 1–6.

[16] IBM, "Quick facts and stats on big data http:http://www.ibmbigdatahub.com/," access February 11, 2016.

[17] K. Kakaes, F. Greenwood, M. Lippincott, S. Dosemagen, P. Meier, and S. Wich, "Drones and aerial observation: New technologies for property rghts, human rights, and global development," Washington, DC, 2015.

[18] J.-N. Lee and K.-C. Kwak, "A trends analysis of image processing in unmanned aerial vehicle," *International Journal of Computer, Information Science and Engineering*, vol. 8, no. 2, pp. 2–5, 2014.

[19] H. Ling, Y. Wu, E. Blasch, G. Chen, H. Lang, and L. Bai, "Evaluation of visual tracking in extremely low frame rate wide area motion imagery," in *Information Fusion (FUSION), 2011 Proceedings of the 14th International Conference on*. IEEE, 2011, pp. 1–8.

[20] B. Liu, Y. Chen, D. Shen, G. Chen, K. Pham, E. Blasch, and B. Rubin, "An adaptive process-based cloud infrastructure for space situational awareness applications," in *SPIE Defense+ Security*. International Society for Optics and Photonics, 2014, pp. 90 850M–90 850M.

[21] K. Liu, B. Liu, E. Blasch, D. Shen, Z. Wang, H. Ling, and G. Chen, "A cloud infrastructure for target detection and tracking using audio and video fusion," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops*, 2015, pp. 74–81.

[22] T. H. Luan, L. Gao, Z. Li, Y. Xiang, and L. Sun, "Fog computing: Focusing on mobile users at the edge," *arXiv preprint arXiv:1502.01815*, 2015.

[23] X. Mei and H. Ling, "Robust visual tracking using l_1 minimization," in *Computer Vision, 2009 IEEE 12th International Conference on*. IEEE, 2009, pp. 1436–1443.

[24] F. Mohammed, A. Idries, N. Mohamed, J. Al-Jaroodi, and I. Jawhar, "Uavs for smart cities: Opportunities and challenges," in *Unmanned Aircraft Systems (ICUAS), 2014 International Conference on*. IEEE, 2014, pp. 267–273.

[25] U. Niethammer, S. Rothmund, and M. Joswig, "Uav-based remote sensing of the slow-moving landslide super-sauze," in *Proceedings of the International Conference on Landslide Processes: From Geomorpholgic Mapping to Dynamic Modelling, Strasbourg, France*, vol. 67, 2009.

[26] K. Palaniappan, F. Bunyak, P. Kumar, I. Ersoy, S. Jaeger, K. Ganguli, A. Haridas, J. Fraser, R. M. Rao, and G. Seetharaman, "Efficient feature extraction and likelihood fusion for vehicle tracking in low frame rate airborne video," in *Information fusion (FUSION), 2010 13th Conference on*. IEEE, 2010, pp. 1–8.

[27] S. Roy, R. Bose, and D. Sarddar, "A fog-based dss model for driving rule violation monitoring framework on the internet of things," *International Journal of Advanced Science and Technology*, vol. 82, pp. 23–32, 2015.

[28] X. Shi, H. Ling, E. Blasch, and W. Hu, "Context-driven moving vehicle detection in wide area motion imagery," in *Pattern Recognition (ICPR), 2012 21st International Conference on*. IEEE, 2012, pp. 2512–2515.

[29] B. Tang, Z. Chen, G. Hefferman, T. Wei, H. He, and Q. Yang, "A hierarchical distributed fog computing architecture for big data analysis in smart cities," in *Proceedings of the ASE BigData & SocialInformatics 2015*. ACM, 2015, p. 28.

[30] P. Tseng, "On accelerated proximal gradient methods for convex-concave optimization. 2008," *Submitted to SIAM J. Optim*, 2009.

[31] R. Wu, Y. Chen, E. Blasch, B. Liu, G. Chen, and D. Shen, "A container-based elastic cloud architecture for real-time full-motion video (fmv) target tracking," in *Applied Imagery Pattern Recognition Workshop (AIPR), 2014 IEEE*. IEEE, 2014, pp. 1–8.

[32] R. Wu, B. Liu, Y. Chen, E. Blasch, H. Ling, and G. Chen, "Pseudo-real-time wide area motion imagery (wami) processing for dynamic feature detection," in *Information Fusion (Fusion), 2015 18th International Conference on*. IEEE, 2015, pp. 1962–1969.

[33] Y. Wu, G. Chen, E. Blasch, L. Bai, and H. Ling, "Feature-based background registration in wide-area motion imagery," in *SPIE Defense, Security, and Sensing*. International Society for Optics and Photonics, 2012, pp. 840 204–840 204.

[34] S. Yi, Z. Hao, Z. Qin, and Q. Li, "Fog computing: Platform and applications," in *Hot Topics in Web Systems and Technologies (HotWeb), 2015 Third IEEE Workshop on*. IEEE, 2015, pp. 73–78.

[35] S. Yi, C. Li, and Q. Li, "A survey of fog computing: concepts, applications and issues," in *Proceedings of the 2015 Workshop on Mobile Big Data*. ACM, 2015, pp. 37–42.

[36] Y. Zheng, L. Capra, O. Wolfson, and H. Yang, "Urban computing: concepts, methodologies, and applications," *ACM Transactions on Intelligent Systems and Technology (TIST)*, vol. 5, no. 3, p. 38, 2014.