Journal of Systems Architecture xxx (xxxx) xxx

Contents lists available at ScienceDirect

Journal of Systems Architecture

journal homepage: www.elsevier.com/locate/sysarc

Light-weight AI and IoT collaboration for surveillance video pre-processing ☆

Yutong Liu [a], Linghe Kong [a,*], Guihai Chen [a], Fangqin Xu [b], Zhanquan Wang [c]

[a] Shanghai Jiao Tong University, Shanghai, China
[b] Shanghai Jianqiao University, Shanghai, China
[c] East China University of Science and Technology, Shanghai, China

ARTICLE INFO

Keywords:
Light-weight AI
IoT collaboration
Wireless surveillance system
Dynamic background modelling
Edge computing

ABSTRACT

As one of the internet of things (IoT) use cases, wireless surveillance systems are rapidly gaining popularity due to their easier deployability and improved performance. Videos captured by surveillance cameras are required to be uploaded for further storage and analysis, while the large amount of its raw data brings great challenges to the transmission through resource-constraint wireless networks. Observing that most collected consecutive frames are redundant with few objects of interest (OoIs), the filtering of these frames before uploading can dramatically relieve the transmission pressure. Additionally, real-world monitoring environment may bring shielding or blind areas in videos, which notoriously affects the accuracy on frame filtering. The collaboration between neighbouring cameras can compensate for such accuracy loss.

Under the computational constraint of edge cameras, we present an efficient video pre-processing strategy for wireless surveillance systems using light-weight AI and IoT collaboration. Two main modules are designed for either fixed or rotated cameras: (i) frame filtering module by dynamic background modelling and light-weight deep learning analysis; and (ii) collaborative validation module for error compensation among neighbouring cameras. Evaluations based on real-collected videos show the efficiency of this strategy. It achieves 64.4% bandwidth saving for the static scenario and 61.1% for the dynamic scenario, compared with the raw video transmission. Remarkably, the relatively high balance ratio between frame filtering accuracy and latency overhead outperforms than state-of-the-art light-weight AI structures and other surveillance video processing methods, implying the feasibility of this strategy.

1. Introduction

Wireless video surveillance systems nowadays perform as guardians in our daily life due to its easier installations and flexible infrastructures [1]. It helps in traffic monitoring [2], parking management, and public security protection in campus, office buildings, or residential communities [3,4]. With the tremendous advancements of artificial intelligence (AI), deep learning models are widely adopted in these applications, which requires real-time video feeding and analysis. However, according to a global forecast report in 2020 [5], the wireless video surveillance systems would enlarge 10.4% usage worldwide from 2020 to 2025, based on the already existed USD 45.5 billion markets. It is apparent that with the growing size of monitoring areas and the number of cameras, the increasing quantity of video streams will bring great challenges on transmission through the resource-constraint wireless network.

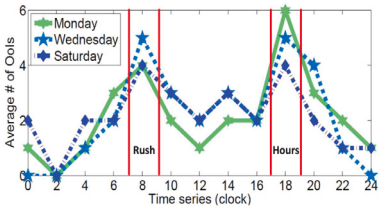
Existing commercial solutions to reduce the transmission quantity on the camera side based on dynamic detection and video coding, which represent temporal and spatial reduction respectively. For the former solution, cameras only start recording and transmitting videos when the different level between two consecutive frames is over the threshold [6]. However, several redundant but dynamic frames will still be captured to the cloud via this method. For example, the public security monitoring in residential communities concerns more about frames containing human activities, animals, and transportation, while frames containing branches sway or garbage bags fly are redundant but dynamic so remained. For the latter solution, video coding standards (e.g. H.264 [7], H.265 [8]) and their variants [9–11] can structurally compress videos without frame filtering. It is compatible with the former solution, which is out of the scope of our discussion.

☆ This paper extends our previous work published in the Proceedings of the International Symposium on Quality of Service (IWQoS) 2019 (Liu et al., 2019).

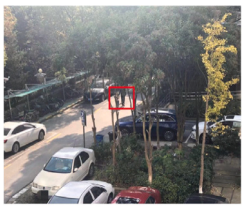
* Corresponding author.
E-mail addresses: isabelleliu@sjtu.edu.cn (Y. Liu), linghe.kong@sjtu.edu.cn (L. Kong), gchen@cs.sjtu.edu.cn (G. Chen), xuqin@gench.edu.cn (F. Xu), zhqwang@ecust.edu.cn (Z. Wang).

[1] We select these three OoI categories in this paper which are frequently appeared and may provide potential safety risks in community life.

https://doi.org/10.1016/j.sysarc.2020.101934
Received 28 June 2020; Received in revised form 10 October 2020; Accepted 3 November 2020
Available online 18 November 2020



(a) The average number of OoIs in different time durations within a day.



(b) The tree shielding on two monitored persons.

Fig. 1. Observations on video redundancy and environmental shielding.

In this paper, we focus on the temporal reduction of the transmitted video quantity. It is worth to notice that there exist a large number of redundant frames in transmitted video streams, which contain less information useful for surveillance applications. Still taking the monitoring in residential communities as an example, we conduct analysis on a 168-hour (one week) monitoring video captured by one camera in a community. As shown in Fig. 1(a), we observe that except the rush hour (7:00–9:00 and 17:00–19:00), there are few OoIs (*i.e.*, human, cars, and animals[1]) detected during other periods in this video. The filtering of these redundant frames will not affect the quality of safety monitoring service but can dramatically save wireless bandwidth during transmission.

Accordingly, we aim to design a content-aware frame filtering strategy to enhance the video compression ratio on the camera side. Well-known deep learning models are powerful tools for content-aware frame filtering, such as SSD [12], YOLO [13], R-CNN [14], etc. But it is not feasible to directly apply these model on edge cameras due to their limited computational capabilities. Methods like DeepMon [15], DeepEye [16], MobileNet [17], and MBBNet [18] propose light-weight model optimization either on model construction or calculation for relatively powerful portable devices like mobile phones. As compared in Table 1, the surveillance cameras have much lower computational capabilities than mobile phones, which require a higher compression degree for deep learning models. In this paper, we discuss the model compression possibility both structurally and computationally. Additionally, massive buildings, plants, and facilities in environments may introduce the shielding even blind monitoring areas in videos (Fig. 1(b)), which would reduce the detection accuracy on redundant frames. To deal with this problem, we add a collaborative validation module based on the edge computing framework [19–21], where neighbouring cameras can help to validate an uncertain frame by peer-to-peer communication.

In a word, we present an efficient video pre-processing in wireless surveillance systems using light-weight AI and IoT collaboration. We consider both static and dynamic surveillance cameras, where dynamic cameras rotate in a stable speed continuously. This strategy is mainly composed of two modules: the frame filtering module on edge cameras and the collaborative validation module among neighbouring pairs. For the first module, we firstly model the background considering the angle of cameras with a proposed DCS-LBP operator and select key frames after background pruning. A light-weight SSD model optimized by channel pruning and convolution acceleration is then applied to these key frames on OoI detection. Only groups of pictures (GOPs) corresponding to key frames which are OoI contained will be transmitted to the cloud. The second module is complementary to the accuracy loss in the first module. The uncertain frames which contain partly shielded objects will be broadcasted to neighbouring cameras for validation.

A surveillance system prototype is implemented from edge cameras to the central server for evaluation. Three main factors are evaluated on real-collected videos to show the efficiency of this strategy: compression ratio, accuracy, and latency. Our method achieves 64.4% compression ratio for videos with a static background and 61.1% for ones with a dynamic background, which dramatically releases the transmission burden. As for accuracy, it successfully filters out 92.6% redundant frames. And for latency, it only causes maximal 0.049 s computational latency on deep learning module and 2.79 s processing delay on validation module for each frame. Its overall balance ratio on accuracy and latency outperforms than state-of-the-art surveillance video processing methods, indicating the satisfactory of accuracy performance and the acceptance of latency overhead.

Table 1
The comparisons on different embedded processors.

Device	CPU	Speed	Energy
Surveillance camera	M0	0.9DMIPS/MHz	85 μW/MHz
	M4	1.25DMIPS/MHz	104 μW/MHz
Mobile phone	A9MP	2000DMIPS/MHz	0.24 W

To sum up, this paper makes the following contributions:

1. **Adaptive background modelling:** To adapt our strategy on most kinds of surveillance cameras, we consider adaptive background modelling for the flexible key frame selection.
2. **Joint model optimization:** We explore the possibility to optimize both the structure and the computation steps to build a light-weight deep learning model, resulting in faster speed and fewer computational consumptions.
3. **Collaborative validation on cameras:** We novelly design a collaborative scheme among edge cameras for validating uncertain frames. It behaves as the compensation for accuracy loss caused by environment shielding.

The remaining part of the paper is organized as follows: Section 2 surveys the related work on two modules in our strategy. Section 3 describes the problem statement and the designed framework. Sections 4 and 5 introduce the details of the design and the prototype implementation, respectively. Section 6 reports the evaluation results. Finally, Section 7 concludes the paper and reviews some future work.

2. Related work

In smart surveillance systems, cameras on the edge have their own computational capabilities and connected to the cloud by network communication, which builds a typical edge computing scheme [19]. The core idea of this scheme is to bring network functions, contents and resources closer to end devices (e.g., edge cameras in surveillance systems) [22], which makes it possible for pre-processing on videos and collaborative validation on the edge before transmission. In this section, we will discuss recent advances on local pre-processing of surveillance videos, light-weight deep learning model optimization, and multi-camera collaboration.

2.1. Surveillance video pre-processing on edge cameras

The widely-used surveillance video pre-processing depends on motion detection [6] or sampling rate deduction [23]. Although these methods keep the lowest computational overhead, they are not precise enough for fine-tuned redundancy deduction, as redundant objects cannot be detected and filtered out. Recently, deep learning-based content-aware video pre-processing methods gain their great popularity. Users can train their models for practical detection targets and the powerful deep learning framework provides high detection accuracy. Well-known deep learning models for object detection can be divided to two types: (i) region-based models: R-CNN [14], SPP-net [24], Fast R-CNN [25], R-FCN [26], etc; and (ii) end-to-end models: YOLO [13] and SSD [12].

Previously discussed in Table 1, off-the-shelf surveillance cameras have 10^3 times lower computational speed and energy consumption than mobile phones. It means that the above mentioned deep learning models cannot be directly applied on cameras, who have over

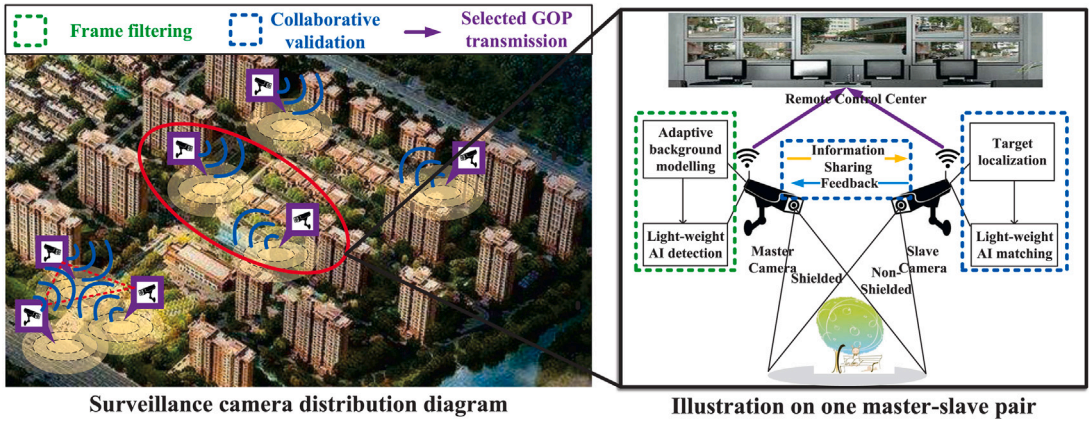


Fig. 2. The overall framework. [Left: Camera distribution in one community (Yellow area implies the monitoring coverage of the camera (static or dynamic), and blue lines implies their WiFi communication); Right: Strategy illustration from edge computing to transmission between one master–slave pair (marked in the red circle on the left).] (For interpretation of the references to colour in this figure legend, the reader is referred to the web version of this article.)

30 layers to process. Light-weight model optimization has been discussed recently. The cache-based convolutional optimization and tensor decomposition in DeepMon [15] can decrease the computational complexity of models. DeepEye [16] leverages the memory caching and SVD-based model compression to support multi-model running. And MobileNet [17] proposes a depth-wise separable convolution, combined by a single-filter derived convolution and 1*1 pointwise convolution. MBBNet [18] utilized channel pruning to reduce the model size, while its input has high resolution and its output is required to be highly accurate for autonomous drive. Besides, according to their evaluations, the feasible devices for these lightweight models should be supported by quite powerful CPU and GPU processors (*e.g.*, 2 GHz), which is not directly applicable for commodity surveillance cameras. In this paper, we tend to optimize the deep learning model from both model construction and calculation aspects, to build a smaller and faster model structure.

2.2. Camera collaboration

Inevitable information missing in uncertain frame detection remains unsolved when using single camera analysis. A scheme called Multi-Camera Coordination and Control (MC^3) [27] can address this issue by master–slave camera collaboration. Collins et al. [28] firstly proposed such a framework, where the master camera is used to track the trajectories of objects, and the slave camera performs specific recognition. Generally speaking, such multi-camera system combines features from different cameras to exploit a scene of an event and can increase the output accuracy by the aggregation of different source of information [29]. Nowadays, it has been leveraged in different scenarios such as surveillance [30,31], sports [32,33], education [34], etc. Related to our paper, the surveillance applications have leveraged this system to increase tracking or detection accuracy. Zhang et al. [30] leveraged object re-identification between cameras for face recognition, while it only considered fixed cameras with pre-known orientations. Jin et al. [31] adopt multi-camera collaboration for pedestrian tracking and proposed a structured vector machine as the cross-camera model, however, the computational latency of this model is non-negligible. To the best of our knowledge, we first utilize multi-camera collaboration for frame validation as an accuracy compensation of light-weight models in video compression. We consider the rotation of cameras to deal with more complex scenario, and seamlessly connect the AI detection module with the validation module, where the previous AI detection results can help to match the projection with detected OoIs, rather than requiring extra decision models.

3. Problem statement and strategy overview

The application scenarios of surveillance cameras include residential communities, offices, campus and so on. In this paper, we take the residential community as an example, where the challenges faced by wireless surveillance system are more representative in its limited area. As shown in the left figure of Fig. 2, surveillance cameras are deployed at important corners, while the type of cameras can be either stable or dynamic. Neighbouring cameras have overlapping monitoring areas at specific timestamps, and they can communicate with the embedded WiFi module in a short range (around 15 metres for commodity cameras) through limited bandwidths. Each camera has two different monitoring mode: active mode and sleep mode. In the active mode, cameras capture videos in a high ratio (*e.g.*, 60 fps), but for the sleep mode is in a low ratio (*e.g.*, 15 fps) [35]. Each camera has its independent processing system with an embedded core processor without GPU, which has the lower computational capability on video streams, and difficult to handle heavy computational tasks (*e.g.*, running a deep learning model with above 10 layers). All these cameras connect direct current and support SD card insertion, where power and storage savings are out of our research scope.

The target of our research is to build an energy-efficient wireless surveillance system by accurately filtering out redundant frames before transmission. So the deduction ratio and the filtering accuracy of redundant frames are two main factors considered in this paper. Considering the limited computational capability of cameras, the computational burden conducted on cameras should be controlled, which can be reflected by acceptable computational latency [36,37]. Taking them as goals, we design a light-weight edge computing strategy for wireless surveillance systems shown in Fig. 2, which mainly composed of two main modules labelled by dotted boxes: frame filtering module and collaborative validation module.

Considering that fewer OoIs will show up other than rush hour in residential communities, each camera works in the sleep mode first and activates until the motion detected for power saving. The first **frame filtering** module aims to filter out redundant frames by light-weight detection on OoIs in edge cameras. Let $\mathrm{F} = \{f_1, f_2, f_3, \ldots, f_{t-1}, f_t, \ldots\}$ denote the current video sequence, where f_t is the frame captured at the timestamp t. For the camera rotated in a fixed angular velocity ω, the period of its rotation can be represented by T. Firstly, we model the dynamic background according to the spatial and temporal information between frames, where the background pixels will be assigned to the value 0 for background pruning. The pruned frames will then perform as the input of key frame selection. In this paper, we select key frames according to the local maximum value of inter-frame differences among consecutive frames, which have better extraction results than selecting

(a) Reference frame (b) Mixed Guassian based foreground (c) SCS-LBP based foreground (d) DCS-LBP based foreground

Fig. 3. Performance visualization of different dynamic background modelling methods.

by orders or threshold according to our experiments. As long as the average inter-frame difference of a frame is a local maximum value, this frame will be selected as a key frame. To remove the noise, we smooth the average difference before calculating the local maximum, which can avoid repetitive selection on similar scenes. Denote the key frame set of video clip F as $K = \{f_{k_1}, f_{k_2}, \dots, f_{k_n}\}$, the GOP is the group of frames between two consecutive key frames, represented as $GOP_i = \{f_{k_i}, f_{k_i+1}, \dots, f_{k_{i+1}}\}$.

Taking these key frames as the input of light-weight deep learning model, three types of objects in frame f_t will be classified as the output: *clear object* with over 50% detection accuracy; *uncertain object* with 0%–50% detection accuracy; and *nothing of interest* with 0% detection accuracy. Correspondingly, the types of key frames are: *selected frame* containing any clear object; *uncertain frame* with only uncertain objects inside; and *redundant frames* without any clear or uncertain objects appeared. Any clear object founded leads to the direct upload of the corresponding GOP. Whenever uncertain objects detected in frames, the **collaborative validation** module will be triggered next.

As shown in the right figure of Fig. 2, a person under the tree is partly shielded, recognized as an uncertain object. Assuming that the master camera is synchronized with all cameras and knows their initial angle and angular velocity at the beginning, the master camera will calculate the nearest camera as its slave pair according to the physical location calculation. Although there is only one slave camera in one-time validation to avoid unnecessary communication overhead, any neighbouring camera in the cluster within a one-hop communication range has a chance to be selected. This uncertain frame and necessary information including the timestamp and the angle of the master camera will be directly forward to the slave camera. Note that each slave node also has functionalities of master cameras, its AI detection results will be extracted for the matching stage to decrease the projection error. After matching, the decision about whether to upload the frame will be sent back to the master camera.

The object detection on each camera will keep running until no OoIs detected in consecutive frames, then this camera will be set back to the sleep mode. Finally, all selected frames will be suppressed by any video coding standard (*e.g.*, MPEG-4, H.264, H.265) and uploaded to the server, which will then be shown in the screen of the remote control room or stored in the server.

4. Detailed designs

In this section, we introduce the detailed designs of two main modules of our pre-processing strategy: frame filtering module and collaborative validation module.

4.1. Frame filtering

The frame filtering module is processed from dynamic background modelling, background pruning, key frame selection, to object detection by optimized deep learning model.

4.1.1. Background modelling and pruning

Considering the change of the background for rotated cameras, it is necessary to adaptively model the background to eliminate the possible error when doing key frame selection at the next step. As the feature vector for modelling the background, we design a DCS-LBP operator extended from SCS-LBP operator [38]. SCS-LBP operator was designed to extract both spatial texture and temporal motion information between consecutive frames. It is defined as:

$$\begin{aligned}\mathrm{SCS-LBP(x,y,t)} = &\sum_{i=0}^{(N/2)-1} s(p_{(i,t)} - p_{(i+N/2)}) \times 2^i \\ &+ f(p_{(x,y,t)} - \overline{\mu}_{(x,y,t-1)}) \times 2^{N/2},\end{aligned} \tag{1}$$

where $p_{(i,t)}$ and $p_{(i+N/2)}$ represents grey levels of centre-sym-metric pixel pairs in current frame f_t. The binary function $s(x)$ is to measure the similarity between these pixel pairs, defined by:

$$s(x) = \begin{cases} 1 & x \geq 0 \\ 0 & \text{otherwise,} \end{cases} \tag{2}$$

and the function $f(t)$ is to assign binary values to either background or foreground, represented as:

$$f(t) = \begin{cases} 0, & \text{if } \left|p_{(x,y,t)} - \overline{\mu}_{(x,y,t-1)}\right| < 2.5 \times \overline{\sigma}_{(x,y,t-1)} \\ 1, & \text{otherwise,} \end{cases} \tag{3}$$

where $\overline{\mu}_{(x,y,t-1)}$ and $\overline{\sigma}_{(x,y,t-1)}$ are estimated mean value and standard deviation of the pixel $p_{(x,y)}$ in previous frame f_{t-1}.

Although the SCS-LBP operator is proper for background modelling for videos, it is built upon the assumption that the video capturing angle is fixed. To deal with the rotation of cameras in our work, we propose Dynamic CS-LBP (DCS-LBP) operator which consider the rotation period of the camera. The specific format of DCS-LBP (abbreviated to D to save paper space) is:

$$\mathrm{D(x,y,t)} = \begin{cases} \sum_{i=0}^{(N/2)-1} A(i,t,N) & t < T \\ \sum_{i=0}^{(N/2)-1} A(i,t,N) + f(p_{(x,y,t)} & \\ \quad - \overline{\mu}_{(x,y,t-T\times\lfloor\frac{t}{T}\rfloor)}) \times 2^{N/2}, & t \geq T, \end{cases} \tag{4}$$

where $A(i,t,N) = s(p_{(i,t)} - p_{(i+N/2)}) \times 2^i$. Define that the camera rotates from the initial angle and back to this angle as one rotation period, with the timeslot T. During the first rotation period, background modelling is performed on individual frames. While in the following replicated period, the frame can always find the corresponding similar background from the first period, rather than its neighbouring frame. The representations of functions $s(x)$ and $f(x)$ is consistent with SCS-LBP. Both $\overline{\mu}_{(x,y,t-T\times\lfloor\frac{t}{T}\rfloor)}$ and $\overline{\sigma}_{(x,y,t-T\times\lfloor\frac{t}{T}\rfloor)}$ are calculated during the first period and used as the reference for following frames.

The dynamic background is modelled by a selective group of K DCS-LBP histograms, represented as $\{d_0, d_1, \ldots, d_{K-1}\}$. And the weight of the kth histogram is w_k, denoting the probability to be the background histogram. We utilize the adaptive background updating method proposed in [38]. Once a new frame arrived, the DCS-LBP histograms for pixels together with their mean values and deviations in this frame will be calculated first. The histograms will then be sorted by a descending order according to the corresponding weight, and the top-K histograms are selected as the prepared background histogram set. The current histograms compare the similarity with the prepared histogram set measured by histogram intersection. If the similarity is below the threshold for all background histograms, the pixel is considered as the foreground, otherwise, the pixel is assigned to 0 for background pruning and the background histogram set is adaptively updated by the highest similarity value. The performance of DCS-LBP when compared with the SCS-LBP and mixed Gaussian model is shown in Fig. 3. It is obvious that the DCS-LBP can present a clearer and cleaner foreground.

4.1.2. Model optimization

As one of the state-of-the-art object detection models, the SSD model [12] outperforms by higher accuracy and detection speed, which is the selection of our work. However, it contains 11 convolutional layers and 8732 prior boxes for each output class, which cause heavy computational burdens for cameras. To efficiently detect redundant frames on surveillance cameras, we optimize the SSD model to a light-weight structure on both structural and computational aspects: model compression and convolutional acceleration.

Model compression. Depending on the sparsity of deep learning model, it can be compressed unstructured [39,40] with higher precise but relying on specific libraries and platforms supports. Conversely, structural compression by channel pruning [41] can fully exploit the redundancy of inter feature maps and directly applied on SSD model without extra libraries required. Besides, this layer-by-layer pruning is suitable to compact single brunch model like the SSD model, which has 11 convolutional layers to be operated.

Two main steps are applied to the target model. Firstly, the most representative channels will be selected, while redundant channels are filtered. Secondly, the selected channels will be reconstructed as the new feature map. For calculation convenience, we denote that a feature map in the SSD model has c channels; the size of convolutional filter W is $n\times c\times k_h\times k_w$; and the size of input volumes X is $N\times c\times k_h\times k_w$, deriving an $N \times n$ output matrix Y. In this denotation, N is the number of samples, and n is the number of output channels, together with $k_h \times k_w$ size of the kernel. The pruning process can be formulated as:

$$\arg\min_{\beta,W} \frac{1}{2N} \left\| Y' - \sum_{i=1}^{c} \beta_i X_i W_i^T \right\|_F^2 \quad (5)$$

$$\text{subject to } \|\beta\|_0 \le c'.$$

Inside of the Frobenius Norm $\|\cdot\|_F$, β_i, X_i, W_i represent the status of channel i (*i.e.*, selected or not selected), new input which pruned channel i from X, and new filters which pruned channel i from W, respectively. Y' is specially designed for the whole model pruning, indicating all outputs from each layer. During this sequential pruning, the accumulated error should be minimized on each output feature maps. So the constraint condition of this formula implies that the retained channels should be fewer than or equal to desired channels, which can be adjusted by the speed-up ratio in training. The optimization of this formula requires two steps. That is, the *channel selection* step can be calculated when W is fixed, while the *feature map reconstruction* step will be further processed when β is fixed. Note that the original code provided by He et al. [42] was specially designed for the GPU-only Caffe environment. We change its settings and test it on a CPU-only environment, and acquire satisfactory results shown in Section 6.

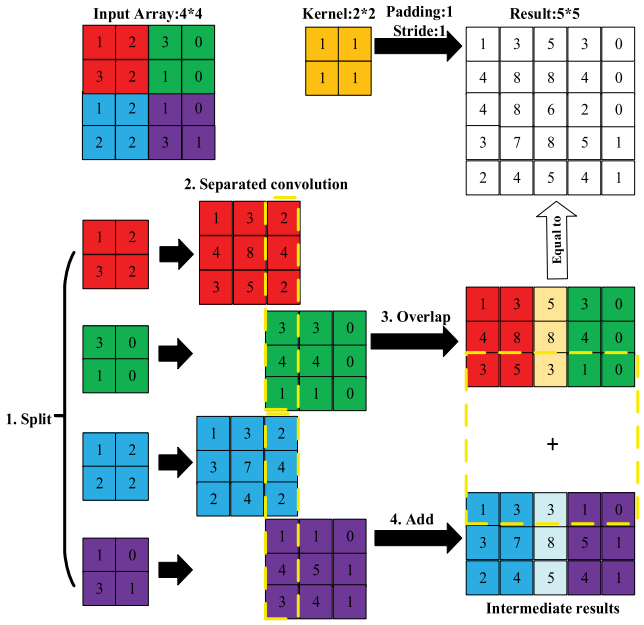


Fig. 4. An example of the OaA algorithm.

Convolutional acceleration. From the computational aspect, convolutional layers are the most time-consuming layers in model running [15]. Especially in SSD model, 11 convolutional layers bring great computational burdens for surveillance cameras. It is necessary to accelerate the convolutional calculation of the SSD model by a simple and fast way.

Taking $M \times M$ and $m \times m$ as the size of inputs and kernels respectively, each kernel takes $O(M^2m^2)$ computational complexity to calculate dot products for a single forward and backward propagation. The fast Fourier transform (FFT) [43] is commonly used for convolutional optimization, because the Hadamard product in the frequency domain is simpler than matrix cartesian product in the space domain. Nonetheless, its complexity $O(M^2 log_2 M)$ is still not satisfied enough for low latency requirement in commodity surveillance systems. For the further reduction on calculation complexity, the overlap-and-add (OaA) optimization technique [44] can promote the convolutional acceleration, holding the complexity of $O(M^2 log_2 m)$. Given an instance, a 128×128 grey level frame with a 2×2 kernel will get the complexity of $O(128^2\times4)$ for standard convolution, $O(128^2\times7)$ for FFT, and $O(128^2\times1)$ for OaA. It implies that the OaA achieves a 1/7 less computational complexity than FFT and a 1/128 less than the standard dot product. Proved from our evaluation results in Section 6, this method further accelerates the running of the SSD model on cameras after structural optimization, which indicates the possibility to speed up deep learning model running from both structural and computational aspects.

Fig. 4 shows a simplified example of 2D grey image processing by OaA, which can be easily extended to 3D colourful images. The grey level frame size in the example is 4×4 and the kernel size is 2×2 with the padding and stride rate 1. Firstly, the frame is divided into 4 blocks with 2×2 size represented by four different colours in Fig. 4, keeping it the same with the kernel size in convolution. The left upper image is the input array with its kernel, while the right upper image is their convolutional result calculated by the standard dot products. The

rest lines illustrate the processing flow of the OaA algorithm, including splitting, separated convoluting, overlapping and adding stages. In the *splitting* stage, four blocks are split from the original input array to four sub-arrays. Each sub-array will be separately convoluted by the same kernel (orange), padding rate and stride with the standard convolution, resulting in four convolutional results. These four results are adjacently added together pair by pair with the overlap rate of $m-1$ (the kernel size generally denoted by $m \times m$), which is 1 in this example. As labelled by yellow dot boxes in Fig. 4, two adjacent convolutional middle results are added on one column while keeping other column concatenated without changing, leading to results shown on the right bottom side. These two further results will then perform overlap and add processes on $m-1$ rows to get the final convolutional result of the input array. It is obvious that the result calculated by OoA is as same as the standard convolutional result given in the first row. During the OoA processing, the separated convolution stage is also computed in the frequency domain for acceleration. The separation in OaA can reduce the convolutional scale, while the overlap and add operations conduct lower computational complexity than dot product.

4.2. Collaborative validation

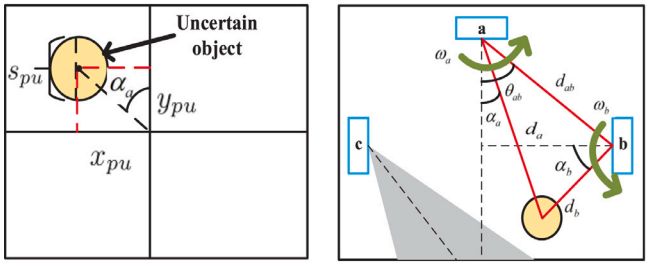
The light-weight deep learning models have lower detection ability than the original model with massive layers. Besides, environmental issues in real surveillance scenarios may also cause potential information loss:

1. **Shielding:** Buildings, plants and facilities are natural barriers for monitoring, especially in modern communities. Concerned objects may be partially or totally shielded by these barriers, driving a lower detection accuracy.
2. **Blind area:** Different from omnidirectional cameras, both static and dynamic cameras have a limited monitoring range. The OoIs which just come into the scene or just leave the scene appear partially, leaving uncertain detection results.

Inspired by the MC^3 scheme [27], we design a collaborative validation strategy among neighbouring cameras to compensate accuracy loss caused by model compression and environmental shielding. Such validation depends on the share of knowledge on target objects within neighbouring nodes built upon the edge computing architecture. The validation is performed locally and communicated within camera pairs. The master camera will trigger the validation module when the uncertain frame is detected. Concretely, the design of this collaborative validation module is shown below.

Due to rotations of cameras, it is quite challengeable to select specific slave camera. In our work, the master camera broadcast validation requests to all neighbouring cameras within one-hop communication range. It implies that all neighbouring cameras have abilities to perform as the slave camera and assist in validation. The validation request sent to slave cameras includes the uncertain frame, the physical location information of requested objects and the corresponding timestamp. Once neighbouring cameras receive the request, it will project the relative locations of these objects in their views and localize them in the corresponding recorded frame. To realize this, we set each camera can store the S-length video clip in its flash memory and refresh it when a new video clip generated, where the length S is slightly larger than the highest broadcast time among its neighbouring camera set. Note that such slave camera selection scheme has lower privacy leakage risk [36] as the frame sharing only happens in a short range (one-hop communication distance).

Depending on the pin-hole imaging theory, we design a linear image projection algorithm for validation target localization. Firstly, the uncertain object u whose detection accuracy is lower than the threshold T ($T = 50\%$ in our experiments) is marked in the box by the former optimized SSD model. As illustrated in Fig. 5(a), its estimated type t_u, centroid location (x_{pu}, y_{pu}), size (maximum occupied pixels



(a) Angle and distance in the frame. (b) Relative position in physical 2-D map.

Fig. 5. Illustrations for collaborative validation calculation.

lengthways) s_{pu} in the frame, and the corresponding timestamp t will be shared to all neighbouring cameras. Its physical location (x_u, y_u) is represented by the distance d_a between object and camera, and the angle α_a is measured between the object and its axle line. We assume that the deployed height and focal of each camera are pre-known. The distance d_a on master camera a is calculated by the pin-hole projection formula [45]:

$$\mathrm{d}_a = \frac{h_u \times f}{s_{pu} + h_a}, \tag{6}$$

where h_u is the practical height of the undefined object u, f is the focal of camera (same for each camera), and h_a is the deployed height of the master camera a. In our experiment, we set the average heights for three detected classes as: 170 cm for human, 20 cm for animals, 150 cm for transportation. So the value of h_u can refer to these settings related to the estimated type t_u.

For the angle calculation, we assume the angle between the centroid of the undefined object and axle line in the frame is the same with the angle between the object and the normal direction of cameras in the physical space. In Fig. 5(a), the angle α_a can be easily calculated by the location of centroid (x_{pu}, y_{pu}) in frame f and transferred as same as the α_a in Fig. 5(b). To imply the east or west of the angle, it will be represented by the positive or negative value for differentiating. Considering the uncertain object is not lying into the monitoring area of camera c at the request timestamp, the camera b is taken as an example which received the above-mentioned information. It needs to calculate the relative angle θ_{ab}, the relative distance d_{ab} with the master camera a, to further calculate the relative angle and distance with the uncertain object for frame localization.

Specifically, the calculation of θ_{ab} depends on the normal direction of their projection areas for camera a and b. Assuming the initial normal directions are pre-known and set to 0, the current normal direction c_i for camera i is calculated by the timestamp t, the rotation period T, and the angular velocity ω_i by:

$$c_i = \omega_i(t - T \times \lfloor \frac{t}{T} \rfloor). \tag{7}$$

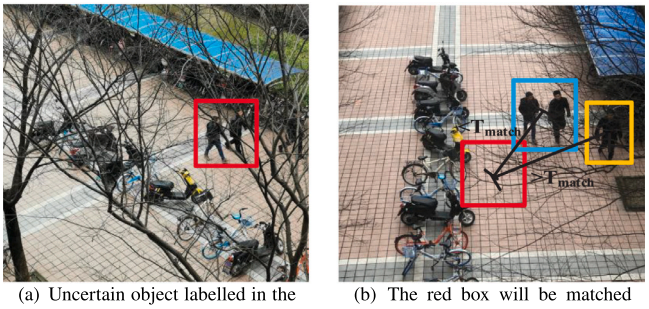
And the relative angle θ_{ab} can be apparently calculated from c_a and c_b. The relative distance d_{ab} is calculated by their fixed physical coordination. Furthermore, the distance d_b for the camera b can be calculated according to the shared information, d_a and α_a,:

$$d_b = \sqrt{(d_a \sin(\theta_{ab} - \alpha_a))^2 + (d_{ab} - d_a \cos(\theta_{ab} - \alpha_a))^2}. \tag{8}$$

And the angle α_b for the camera b can be calculated as:

$$\alpha_b = \arccos \frac{d_{ab} - d_a \cos(\theta_{ab} - \alpha_a)}{d_b}. \tag{9}$$

Similarly, such physical distances and angles can be inversely transferred to the size s_{pb} and the angle α_b in the frame f_{bt} of the camera b at timestamp t according to the reverse function of (6). A $s_{pb} \times s_{pb}$ box will be drawn as the localization of uncertain object u in the camera b.



(a) Uncertain object labelled in the red box.

(b) The red box will be matched with the blue box.

Fig. 6. Localization and matching illustrations. (For interpretation of the references to colour in this figure legend, the reader is referred to the web version of this article.)

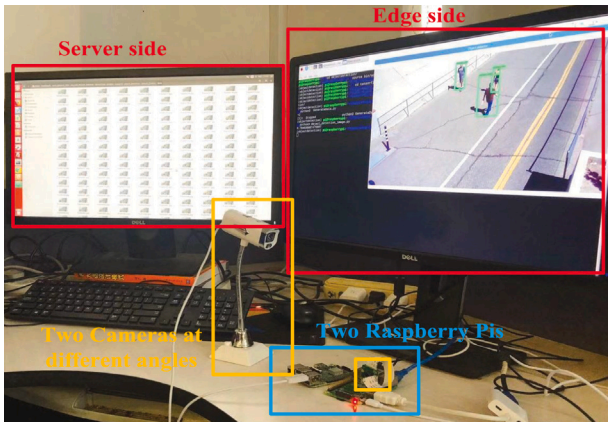


Fig. 7. Prototype illustration. (For interpretation of the references to colour in this figure legend, the reader is referred to the web version of this article.)

However, as the projection from the physical world to pictures will lose parameters in one dimension, this simplified algorithm will bring potential accuracy loss. Correspondingly, we design a *matching mechanism* for error compensation. The frame captured in validating timestamp is input into the optimized SSD model and any OoI will be marked in a box by this model. The matching mechanism will find out the closest detection box with the localized box acquired in the previous step. If the shortest distance between two centroids of regions is not more than threshold T_{match}, we consider these two regions as the same region, where we will match the calculated location with this detected location. The threshold T_{match} is set by the experimental average of multiple measurements. Illustrated in Fig. 6, the shielded humans labelled in the red box of the left picture is localized as the same red box in the second picture by calculation. There are two OoI detection regions labelled in blue and yellow boxes in Fig. 6(b). According to distances between the localized box and detection boxes, the red box is relatively closer to the blue box, and the distance is shorter than the threshold T_{match}. So we will match the red box with the blue box.

After the matching stage, three kinds of results will be given on each camera by the validation module, which are sent as feedbacks to the master camera: (i) Drop, when there is no OoI found from the matched region in cameras; (ii) Confirm, when there is OoI found in the matched region; and (iii) Uncertain, when the OoI in the matched region also has the detection accuracy lower than 50%. Taking all feedbacks from neighbouring cameras into consideration, three kinds of actions will be taken by the master camera accordingly:

1. **Ignore:** When all neighbouring cameras give "drop" feedback, the master camera will not select corresponding GOP for transmission.
2. **Transmit:** When there are any "confirm" feedback, the master camera will directly transmit the GOP.
3. **Transmit but uncertain:** If there are no "confirm" feedback but "uncertain" feedback exists, the master camera will transmit the GOP, leaving it to the server for further checking.

5. Implementation

As shown in Fig. 7, we implement a prototype of the whole surveillance system on two sides: the edge camera side and the server side. On the camera side, we use Raspberry Pi 3bs for experiments (labelled in the blue box), embedded with ARM Cortex-A53 and no GPU. Supported by 802.11n WiFi module and external camera module v2, they perform as surveillance cameras with video capturing, processing and wireless communication functions. To simulate both static and dynamic cameras deployed in neighbouring locations, we select the camera module on the Raspberry Pi as a static one and a commodity external camera with a USB connection as the dynamic one with a fixed rotation speed (labelled in yellow boxes). We utilize the motioneyeos API proposed in [6] as the motion detection function in each camera. It would increase the capturing frame rate for cameras when motion detected while remaining to the sleep mode in other time periods. The frame rate in the sleep mode and the active mode are 15 Hz and 32 Hz, respectively. And we set the capturing resolution as the typical 640*480 dpi.

What is more, the Linux system computer performs as the server, with 64-bit Ubuntu 14.04 LTS version, 8 core 3.6 GHz Intel Core i7 CPU and Kabylake GT2 770 GPU. The detection performance of the light-weight deep learning model is visualized on the Raspberry Pis connected monitor at the right side. And the performance of the original deep model is visualized on the monitor at the left side, which performs as the ground truth for the detection accuracy evaluation.

Both the light-weight SSD model and its original version are trained locally, while on different datasets. As good performances of models also rely on the high-quality training set, we utilize MS COCO dataset [46] published by Microsoft, because of its great training results. As mentioned before, there will be 8732 prior box results for each detected class in the SSD model, while 90 classes in MS COCO dataset require a large number of calculations. From our observations, some objects in residential communities have no need to be detected even it is dynamic, such as plants (*e.g.*, trees, grass, flowers) and buildings. While for other facilities like doors, mailboxes, and exercise facilities, etc., we consider that their movements are involved in human activities. So as long as we detect humans in frames, these facilities will also be included in selected frames, with no need for extra detection. At last, we focus seven categories in MS COCO dataset: human, dog, cat, car, bicycle, bike, and bus (especially for campus scenario), which are frequently appeared in captured videos and have potential risks in our daily life. Additionally, these selected categories have similar features: dog and cat are all animals with four feet and small shapes; car, bicycle, bike, and bus are transportation tools with wheels. So we combine these similar categories into three kinds to reduce its running time on surveillance cameras: human, transportation, and animal. We filter out and summarize the original image set and XML annotation file of MS COCO dataset, and finally acquire 8462 samples for people, 10 190 samples for transportation, and 3759 samples for animals in total. The light-weight model is then trained on this processed MS COCO dataset. The 10% of total samples will perform as testing samples while another 90% samples perform as training samples. It is trained for light-weight models on Raspberry Pis. But on the server side, the original SSD model is trained on the original MS COCO dataset as the ground truth.

For the collaborative validation module, the master and the slave camera will first share their location information, where the relative distances and angles can be measured. We realize this module by writing approximately 200 lines of Python codes on master and slave

Table 2
Compression ratio comparisons for both static and dynamic cameras.

Feature	Static	Dynamic
Original	325M	349M
MPEG-4	133M(59.1%)	189.4M(45.8%)
Litedge	96.8M(70.2%)	123.2M(64.7%)
Ours	115.7M(64.4%)	135.6M(61.1%)

nodes, including information broadcasting, projection calculation, and matching. To avoid packet loss, all information sharing between them are in multiple times until getting the "received" feedback.

6. Evaluation

To prove the efficiency of the proposed strategy, experiments are conducted on real-collected videos. Two kinds of videos are captured by either static or dynamic cameras in the same residential communities simultaneously. Three main indicators are evaluated in this section. At the beginning, the compression ratios (frame deduction ratio) for both static and dynamic cameras are present in Section 6.1, which is the main target of this paper. Secondly, Section 6.2 evaluates the processing latency on both calculation and transmission. Additionally, Section 6.3 evaluates the accuracy performances, including the detection accuracy, the validation compensation ratio, and the overall filtering accuracy. The tradeoff between latency and accuracy is also an important indicator of the performance, which is finally discussed by comparisons with controlled experiments and related works in Section 6.4.

6.1. Compression ratio evaluation

Methodology. Two video traces collected from the same residential community by both static and dynamic cameras are utilized for compression ratio comparisons. Three benchmarks are introduced to compare with our strategy:

1. Original: This is the baseline of the comparison, without any compression on the video.
2. MPEG-4 [47]: MPEG-4 is a content-aware video coding standard, which separates the video objects as the foreground and describes them by motion, shape, and texture information. The other pixels are background, which has a higher compression ratio than the foreground.
3. Litedge [48]: Litedge is based on light-weight deep learning model for content-aware frame filtering, while it detects frame-by-frame with a constant sampling ratio.

The video size Q_{com} compressed by these four methods are recorded to calculate the compression ratio R with the original video size Q_{ori} by:

$$R = \frac{Q_{ori} - Q_{com}}{Q_{ori}} \times 100\% \tag{10}$$

Result. Table 2 summarizes compression ratios for four methods on both static and dynamic cameras. It is same for all compression methods that the compression ratio for videos with a dynamic background is lower than the static background, because of the richer information contained in it. Our method has a similar compression ratio with Litedge but slightly lower because of our GOP transmission strategy. Compared with the frame selection and transmission in Litedge, the GOP transmission can avoid large distortion for surveillance applications, especially when trajectories or motion status of objects are concerned. Note that our method is compatible with coding standards like MPEG-4, the compression ratio can be lower than the number recorded in this table in real deployments.

6.2. Latency evaluation

Methodology. In the latency evaluation, we aim to answer the following three questions in latency evaluations:

1. How much calculation latency can be saved by the optimized SSD model?
2. How much processing latency is further introduced by the collaborative validation?
3. How much transmission latency can be saved by the proposed scheme?

To answer the first question, we select key frames from both static and dynamic videos and aggregate them as inputs for the optimized and original SSD models. The processing time for each frame will be recorded and drawn as a CDF figure. For the second question, uncertain frames acquired from the optimized SSD model will be input into the collaborative validation module for processing latency measurement. And for the third question, the overall transmission latency is measured with a 100k/s uploading speed from the camera to the server. Both the validation latency and the transmission latency are extracted from the log file.

Result. Fig. 8(a) shows the CDF of the processing latency for the optimized SSD model on each frame, compared with the original model running on cameras. We observe that the average latency is 0.049 s for the optimized model, while it is 0.06 s for the original model, leading to the around 18.3% deduction. It proves that our optimization method can help to accelerate model running and also fit the model into light-weight devices which have limited computational capabilities.

As recorded by the log file, the average latency for the collaborative validation module is 2.79 s. It is good to see that the matching stage among neighbouring cameras is performed in a distributed way, where only the highest matching time affects the overall validation latency, rather than the total delay.

According to the real measured transmission latency, we found that when the transmission speed keeps the same, the transmission latency is directly related to the sizes of videos. So the transmission deduction tendency is consistent with the compression ratio shown in Table 2.

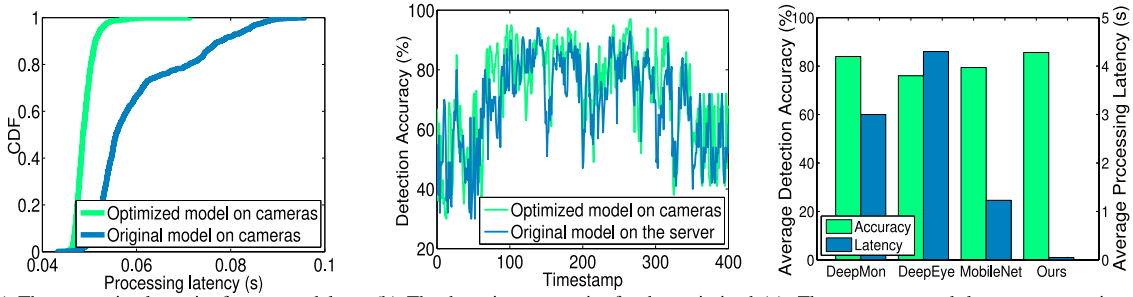
Finally, we calculate the real-time latency for each frame processed in situ, starting from its capturing, to processing on cameras, to considering for transmission, until its arrival to the server. It is calculated through the total latency recorded for a video clip divided by its number of frames. And the average result shows 0.27 s for each frame. Under a 32 Hz capturing rate, the camera can easily handle such processing stage with a 3M buffer.

6.3. Accuracy evaluation

Methodology. To measure the filtering accuracy on the optimized model, we first run the optimized model on the camera side, with the original model running on the server as the ground truth. Taking the selected key frame set as the input, the accuracy difference between them implies two conditions: frames are failed to be filtered or mistakenly considered as redundancy. The first kind of frames becomes the compensation target in collaborative validation module. As the results are quite stable after over 500 attempts, we only show accuracy results for the former 500 key frames in result figures.

We further define the compensation level to evaluate the performance of collaborative validation module. We denote the redundant frames founded by light-weight AI as f_{AI}; the newly detected redundant frames in validation module as f_{val}; and the number of real redundant frames detected by the original model running on PC as f_{real}. Then, the final accuracy Acc can be represented by this compensation ratio R_{com}:

$$Acc = \frac{f_{AI}}{f_{real}} + R_{com} = \frac{f_{AI} + f_{val}}{f_{real}} \tag{11}$$



(a) The processing latencies for two models on edge cameras.

(b) The detection accuracies for the optimized model and the original model.

(c) The accuracy and latency comparisons with other light-weight AI methods.

Fig. 8. Evaluations on latencies, accuracies, and their tradeoff.

Table 3
The summarization on the overall tradeoff comparisons.

Method	Original		Filtering-Only		Litedge [48]		Zhang et al. [30]		Ours	
Indicators	Accuracy	Latency	Accuracy	Latency	Accuracy	Latency	Accuracy	Latency	Accuracy	Latency
Results	93%	676 s	82.3%	301.62 s	89.23%	372.86 s	90%	329.88 s	92.6%	332.9 s
Balance ratio	0.138		0.273		0.239		0.273		**0.278**	

Results. As shown in Fig. 8(b), the optimized model running on our prototype has a quite similar accuracy with the original model running on PC. The average frame detection accuracy on the optimized model is 85.6% and 90.2% on the original SSD model. Besides, the compensation ratio is $R_{com} = 7\%$, leading to the 92.6% overall accuracy on redundant frame detection of our method. It illustrates the improvement provided by the IoT collaboration module.

6.4. Tradeoff evaluation

Methodology. The tradeoff between accuracy and latency can comprehensively imply the efficiency of the proposed scheme. Key frames selected from videos with static and dynamic backgrounds are aggregated as inputs for all methods. We first evaluate the tradeoff of the light-weight AI part, compared with three popular light-weight AI structures: DeepMon [15], DeepEye [16], and MobileNet [17]. We then evaluate the tradeoff of the overall scheme with controlled experiments and the related work:

1. Original transmission: Its accuracy is measured when running the original SSD model on the server, performing as the baseline in the comparison.
2. Filtering-only: Its accuracy is measured when running the light-weight SSD model on cameras. This method is a controlled experiment where only the frame filtering module works.
3. Litedge [48]: This method is designed to be suitable only for static surveillance scenario and detect frame-by-frame with a constant sampling ratio.
4. Zhang et al. [30]: This is a closely related work proposed in the edge computing scheme. It assigns priorities of frames by vision-based object detection algorithms and schedules them for transmission in controllers.

The latencies of all these methods are measured from video recording to receiving on the server. Considering that higher accuracy and lower latency illustrates better performance, we simply divide the accuracy with latency to represent the *balance ratio* as the indicator of the tradeoff.

Result. The tradeoff of light-weight AI part is shown in Fig. 8(c), our design achieves a better balance between detection accuracy and latency. Specifically, the balance ratio of DeepMon is 27.98, 18.1 for DeepEye, 65.55 for MobileNet, and 1746.9 for our optimization. Table 3 implies the feasibility of our light-weight AI design and the possibility of the joint optimization on model structure and computation. Some observations can be noticed from respective comparisons. At first, the accuracy after adding the collaborative validation module is increased, and the higher balance ratio of our strategy shows that the extra latency overhead is acceptable. Secondly, compared with Litedge, it is apparent that our method performs better when processing frames with dynamic backgrounds. At last, our method has higher accuracy than Zhang's method [30], depending on the powerful deep learning tool. Due to the frame scheduling strategy designed on the controller in their method, the frames aggregated to controllers cause non-negligible latency overhead, which reduces their balance ratio. On the contrary, our extra computational latency introduced by IoT collaboration has relatively lower side effect for our balance ratio.

7. Summary and conclusion

This work is based on the observation that there exists a large number of redundant frames in surveillance videos which contain less information for surveillance applications. Spatially, simply reducing the sampling ratio or filtering frames by dynamic detection in commercial cameras still have room to enhance the compress ratio. To further reduce the transmission burden of surveillance videos via wireless spectrum, in this paper, we present an efficient video pre-processing scheme composed of light-weight AI and IoT collaboration. Respectively, it solves two main problems in AI-based frame filtering: (a) The limited computational capability of surveillance cameras; (b) The accuracy loss caused by model compression and environmental shielding.

Our method considers both static or dynamic surveillance scenarios. After dynamic background modelling by the novel DCS-LBP operator, the redundant GOPs in raw videos are then filtered by content-aware key frame selection on edge cameras, which is realized by a light-weight SSD model to fit their limited computational capabilities. The object detection model is optimized by channel pruning and convolutional acceleration via OaA method.

The accuracy loss caused by model shrinkage and environmental shielding is compensated by a collaborative validation module among neighbouring cameras within one-hop range. The uncertain target is localized and re-identified by the slave camera and provide feedback to the master camera for uploading decision making.

Evaluations based on real-collected videos prove the feasibility of this strategy. It is shown that our method can dramatically reduce the transmission quantity, together with a relatively high balance ratio between system accuracy and latency among controlled experiments and state-of-the-art methods.

To further increase the utility of our method, we highlight several interesting future directions. On the one hand, considering that more

smart types of cameras are presented in the market, such as tracking-driven cameras [49] whose rotation speed is not fixed while followed with targets, the rotation-related discussion in this paper should be more flexible when dealing with this scenario. On the other hand, more efficient model compression methods can be explored especially when there exist strong spatial and temporal correlations among surveillance cameras. For example, distributed learning [50] or teacher–student knowledge distillations [51] can help to further accelerate or compress the model.

Declaration of competing interest

One or more of the authors of this paper have disclosed potential or pertinent conflicts of interest, which may include receipt of payment, either direct or indirect, institutional support, or association with an entity in the biomedical field which may be perceived to have potential conflict of interest with this work. For full disclosure statements refer to https://doi.org/10.1016/j.sysarc.2020.101934.

Shanghai Jiao Tong University, Shanghai, China.
Shanghai Jianqiao University, Shanghai, China.
East China University of Science and Technology, Shanghai, China.

Acknowledgements

This work was supported in part by National Key R&D Program of China 2018YFB1004703, NSFC, China grant 61972253, 61672349, U190820096, 61672348, 61672353, the Program for Professor of Special Appointment (Eastern Scholar) at Shanghai Institutions of Higher Learning, China.

References

[1] R. Brzoza-Woch, A. Ruta, K. Zielinski, Remotely reconfigurable hardware-software platform with web service interface for automated video surveillance, J. Syst. Archit. 59 (7) (2013) 376–388.
[2] A. Appathurai, R. Sundarasekar, C. Raja, E.J. Alex, C.A. Palagan, A. Nithya, An efficient optimal neural network-based moving vehicle detection in traffic video surveillance system, Circuits Syst. Signal Process. 39 (2) (2020) 734–756.
[3] W.N.I. Al-Obaydy, S.A. Suandi, Automatic pose normalization for open-set single-sample face recognition in video surveillance, Multimedia Tools Appl. 79 (3–4) (2020) 2897–2915.
[4] S.V. Venkatesh, A.P. Anand, G.S. S., A. Ramakrishnan, V. Vijayaraghavan, Real-time surveillance based crime detection for edge devices, in: G. M. Farinella and P. Radeva and J. Braz (Eds.), VISIGRAPP, Valletta, Malta, 2020, pp. 801–809.
[5] Video Surveillance Market Global Forecast to 2025, https://www.marketsandmarkets.com/Market-Reports/video-surveillance-market-645.html.
[6] ccrisan, MotionEyeOS, https://github.com/ccrisan/motioneyeos.
[7] T. Wiegand, G.J. Sullivan, G. Bjøntegaard, A. Luthra, Overview of the h.264/AVC video coding standard, IEEE Trans. Circuits Syst. Video Technol. 13 (7) (2003) 560–576.
[8] V. Sze, M. Budagavi, G.J. Sullivan (Eds.), High Efficiency Video Coding (HEVC), Algorithms and Architectures, in: Integrated Circuits and Systems, Springer, 2014.
[9] Y. Tsai, M. Liu, D. Sun, M. Yang, J. Kautz, Learning binary residual representations for domain-specific video streaming, in: AAAI, New Orleans, Louisiana, USA, 2018, pp. 7363–7370.
[10] A. Serrano, E. Garces, B. Masiá, D. Gutierrez, Convolutional sparse coding for capturing high-speed video content, Comput. Graph. Forum 36 (8) (2017) 380–389.
[11] L. Kong, R. Dai, Efficient video encoding for automatic video analysis in distributed wireless surveillance systems, TOMCCAP 14 (3) (2018) 72:1–72:24.
[12] W. Liu, D. Anguelov, D. Erhan, C. Szegedy, S.E. Reed, C. Fu, A.C. Berg, SSD: Single shot multibox detector, in: ECCV, Amsterdam, the Netherlands, 2016, pp. 21–37.
[13] J. Redmon, A. Angelova, Real-time grasp detection using convolutional neural networks, in: 2015 IEEE International Conference on Robotics and Automation (ICRA), IEEE, 2015, pp. 1316–1322.
[14] R.B. Girshick, J. Donahue, T. Darrell, J. Malik, Rich feature hierarchies for accurate object detection and semantic segmentation, in: IEEE CVPR, Columbus, OH, USA, 2014, pp. 580–587.
[15] H.N. Loc, Y. Lee, R.K. Balan, DeepMon: Mobile gpu-based deep learning framework for continuous vision applications, in: MobiSys, Niagara Falls, NY, USA, 2017, pp. 82–95.
[16] A. Mathur, N.D. Lane, S. Bhattacharya, A. Boran, C. Forlivesi, F. Kawsar, Deepeye: Resource efficient local execution of multiple deep vision models using wearable commodity hardware, in: MobiSys, Niagara Falls, NY, USA, 2017, pp. 68–81.
[17] A.G. Howard, M. Zhu, B. Chen, D. Kalenichenko, W. Wang, T. Weyand, M. Andreetto, H. Adam, Mobilenets: Efficient convolutional neural networks for mobile vision applications, 2017, arXiv preprint arXiv:1704.04861.
[18] Z. Ouyang, J. Niu, T. Ren, Y. Li, J. Cui, J. Wu, Mbbnet: An edge IoT computing-based traffic light detection solution for autonomous bus, J. Syst. Archit. 109 (2020) 101835.
[19] G. Jia, G. Han, H. Rao, L. Shu, Edge computing-based intelligent manhole cover management system for smart cities, IEEE Internet Things J. 5 (3) (2018) 1648–1656.
[20] C. Dou, S. Zhang, H. Wang, L. Sun, Y. Huang, W. Yue, Adhd fmri short-time analysis method for edge computing based on multi-instance learning, J. Syst. Archit. 111 (2020) 101834.
[21] J. Zhang, M. Ma, W. He, P. Wang, On-demand deployment for IoT applications, J. Syst. Archit. (2020) 101794.
[22] S. Wang, X. Zhang, Y. Zhang, L. Wang, J. Yang, W. Wang, A survey on mobile edge networks: Convergence of computing, caching and communications, IEEE Access 5 (2017) 6757–6779.
[23] Y. Zhang, H. Wang, D. Zhao, Up-sampling dependent frame rate reduction for low bit-rate video coding, in: DCC, Snowbird, Utah, USA, 2011, pp. 489.
[24] K. He, X. Zhang, S. Ren, J. Sun, Spatial pyramid pooling in deep convolutional networks for visual recognition, in: ECCV, Zurich, Switzerland, 2014, pp. 346–361.
[25] R.B. Girshick, Fast R-CNN, in: IEEE ICCV, Santiago, Chile, 2015, pp. 1440–1448.
[26] J. Dai, Y. Li, K. He, J. Sun, R-FCN: Object detection via region-based fully convolutional networks, in: NeurIPS, Barcelona, Spain, 2016, pp. 379–387.
[27] P. Natarajan, P.K. Atrey, M.S. Kankanhalli, Multi-camera coordination and control in surveillance systems: A survey, TOMCCAP 11 (4) (2015) 57:1–57:30.
[28] R.T. Collins, A.J. Lipton, H. Fujiyoshi, T. Kanade, Algorithms for cooperative multisensor surveillance, Proc. IEEE 89 (10) (2001) 1456–1477.
[29] A.S. Olagoke, H. Ibrahim, S.S. Teoh, Literature survey on multi-camera system and its application, IEEE Access 8 (2020) 172892–172922.
[30] T. Zhang, A. Chowdhery, P. Bahl, K. Jamieson, S. Banerjee, The design and implementation of a wireless video surveillance system, in: ACM/IEEE Mobicom, Paris, France, 2015, pp. 426–438.
[31] Z. Jin, L. An, B. Bhanu, Group structure preserving pedestrian tracking in a multicamera video network, IEEE Trans. Circuits Syst. Video Technol. 27 (10) (2016) 2165–2176.
[32] P. McIlroy, Hawk-eye: Augmented reality in sports broadcasting and officiating, in: IEEE/ACM International Symposium on Mixed and Augmented Reality, IEEE Computer Society, 2008, p. xiv.
[33] J. Li, J. Xu, F. Zhong, X. Kong, Y. Qiao, Y. Wang, Pose-assisted multi-camera collaboration for active object tracking, in: AAAI, Vol. 34, 2020, pp. 759–766.
[34] L. Mothwa, J.-R. Tapamo, T. Mapati, Conceptual model of the smart attendance monitoring system using computer vision, in: SITIS, IEEE, 2018, pp. 229–234.
[35] T. Rault, A. Bouabdallah, Y. Challal, Energy efficiency in wireless sensor networks: A top-down survey, Comput. Netw. 67 (2014) 104–122.
[36] Z. Shao, C. Xue, Q. Zhuge, M. Qiu, B. Xiao, E.-M. Sha, Security protection and checking for embedded system integration against buffer overflow attacks via hardware/software, IEEE Trans. Comput. 55 (4) (2006) 443–453.
[37] M. Qiu, Z. Jia, C. Xue, Z. Shao, E.H.-M. Sha, Voltage assignment with guaranteed probability satisfying timing constraint for real-time multiproceesor DSP, J. VLSI Signal Process. Syst. Signal Image Video Technol. 46 (1) (2007) 55–73.
[38] G. Xue, J. Sun, L. Song, Dynamic background subtraction based on spatial extended center-symmetric local binary pattern, in: IEEE ICME, Singapore, 2010, pp. 1050–1054.
[39] S. Han, H. Mao, W.J. Dally, Deep compression: Compressing deep neural networks with pruning, trained quantization and huffman coding, 2015, arXiv preprint arXiv:1510.00149.
[40] X. Liu, J. Pool, S. Han, W.J. Dally, Efficient sparse-winograd convolutional neural networks, 2018, arXiv preprint arXiv:1802.06367.
[41] Y. He, X. Zhang, J. Sun, Channel pruning for accelerating very deep neural networks, in: IEEE ICCV, Venice, 2017, pp. 1398–1406.
[42] X.Z. Yihui He, J. Sun, Channel pruning github code, https://github.com/yihui-he/channel-pruning.
[43] M. Mathieu, M. Henaff, Y. LeCun, Fast training of convolutional networks through ffts, 2013, arXiv preprint arXiv:1312.5851.
[44] T. Highlander, A. Rodriguez, Very efficient training of convolutional neural networks using fast fourier transform and overlap-and-add, in: BMVC, Swansea, UK, 2015, pp. 160.1–160.9.
[45] wikipedia, Pin-hole projection formula, https://en.wikipedia.org/wiki/Pinhole_camera_model.
[46] T. Lin, M. Maire, S.J. Belongie, J. Hays, P. Perona, D. Ramanan, P. Dollár, C.L. Zitnick, Microsoft COCO: Common objects in context, in: ECCV, Zurich, Switzerland, 2014, pp. 740–755.
[47] J.L. Mitchell, W.B. Pennebaker, C.E. Fogg, D.J. Legall, MPEG Video Compression Standard, 99 (5) (1996) 1666–1675.

[48] Y. Liu, L. Kong, M. Hassan, L. Cheng, G. Xue, G. Chen, Litedge: towards light-weight edge computing for efficient wireless surveillance system, in: ACM IWQoS, Phoenix, AZ, USA, 2019, pp. 31:1–31:10.
[49] P.D.Z. Varcheie, G. Bilodeau, Human tracking by IP PTZ camera control in the context of video surveillance, in: M. S. Kamel and A. C. Campilho (eds.), Springer ICIAR, Vol. 5627, Halifax, Canada, 2009, pp. 657–667.
[50] O. Gupta, R. Raskar, Distributed learning of deep neural network over multiple agents, J. Netw. Comput. Appl. 116 (2018) 1–8.
[51] M.-C. Wu, C.-T. Chiu, Multi-teacher knowledge distillation for compressed video action recognition based on deep learning, J. Syst. Archit. 103 (2020) 101695.

Yutong Liu is currently a Ph.D. candidate in Computer Science with Shanghai Jiao Tong University. In 2017, she received her B. E. degree in Computer Science and Technology from Ocean University of China. Her research interests include wireless sensing, artificial intelligence, and mobile computing.

Linghe Kong is currently a research professor in Shanghai Jiao Tong University. Before that, he was a postdoctoral fellow at Columbia University and McGill University. He received his Ph.D. degree in Computer Science with Shanghai Jiao Tong University 2012, Master degree in Telecommunication with TELECOM SudParis 2007, B. E. degree in Automation with Xidian University 2005. His research interests include wireless communications, sensor networks and mobile computing.

Guihai Chen earned his B.S. degree from Nanjing University in 1984, M.E. degree from Southeast University in 1987, and Ph.D. degree from the University of Hong Kong in 1997. He is a distinguished professor of Shanghai Jiaotong University, China. He had been invited as a visiting professor by many universities including Kyushu Institute of Technology, Japan in 1998, University of Queensland, Australia in 2000, and Wayne State University, USA during September 2001 to August 2003. He has a wide range of research interests with focus on sensor network, peer-to-peer computing, high performance computer architecture and combinatorics.

Fangqin Xu is a profess or of Shanghai Jianqiao University. In 2018, he graduated from communication and information system major of East China Normal University, and his master's degree from computer technology major of Huazhong University of science and technology. His main research direction is Internet of things technology application and computer application technology.

Zhanquan Wang is an associate professor at Department of Computer Science and Engineering of East China University of Science and Technology in Shanghai, China. He received his Ph.D. in computer science from Zhejiang University in China in 2005. His current research interests include Spatial Datamining, GIS, Educational technology.