IEEE Access
Multidisciplinary ⋮ Rapid Review ⋮ Open Access Journal

Received May 2, 2016, accepted May 13, 2016, date of publication May 24, 2016, date of current version June 13, 2016.

Digital Object Identifier 10.1109/ACCESS.2016.2572121

Moving Object Counting Using a Tripwire in H.265/HEVC Bitstreams for Video Surveillance

YUNG-WEI CHEN[1], KAI CHEN[1], SHIH-YI YUAN[2], AND SY-YEN KUO[1], (Fellow, IEEE)
[1]Department of Electrical Engineering, National Taiwan University, Taipei 10802, Taiwan
[2]Department of Communication Engineering, Feng Chia University, Taichung 40250, Taiwan

Corresponding author: Y.-W. Chen (francis2001tw@gmail.com)

This work was supported by the Ministry of Science and Technology, Taiwan, under Grant NSC 102-2221-E-002-136-MY3.

ABSTRACT The objective of this paper is to estimate the number of moving objects that passes through a specific area without fully decoding the H.265/high-efficiency video coding (HEVC) bitstreams. First, the foreground prediction blocks are extracted according to the motion vectors of the H.265/HEVC bitstreams. Next, these foreground prediction blocks are clustered into the region of interests (ROIs), which are the possible area position of moving objects in the current frame. Finally, the state of moving objects is identified by matching moving objects and these ROIs. In order to estimate the number of moving objects, which move toward a pre-defined direction, a tripwire is set to a detecting area. Any moving objects crossing the tripwire and satisfying the intrusion conditions are counted. With the proposed method, the number of moving objects can be directly estimated in the compressed domain video. This approach significantly increase the processing speed more than 400% at the cost of less than 0.02% accuracy degradation compared with the traditional pixel domain approach. The research results can be applied to traffic management, real-time analysis of surveillance application, and other related areas.

INDEX TERMS H.265/HEVC, object counting, video surveillance applications.

I. INTRODUCTION

With the development of intelligent video surveillance system, moving object counting has been achieved by image analysis. In order to estimate the number of moving objects stored in the video recorder system, most of the existing intelligent surveillance systems need to analyze video after fully decoding bitstreams. Such approaches result in heavy computational cost on the surveillance system, especially with high quality and large amount video streams. Therefore, it can be of great help to estimate these objects without fully decoding the bitstreams in performing real-time video analysis.

Over the last 10 years, the mainstream specification of video streaming is H.264/AVC. With the publication of H.265/HEVC as the next generation of video streaming standard in April 2013 [1], image analysis based on H.265/HEVC is predicted to be the main trend of the security industry development [2]. Therefore, this paper proposes an approach to estimate the number of moving objects without the need to fully decode the H.265/HEVC bitstreams. Therefore, this approach is suitable for future intelligent video surveillance systems.

Traditional intelligent video surveillance systems mainly analyze the pixel domain image to detect moving objects. Chen *et al.* [3] presented a video surveillance system process pipeline that has four stages: 1) FG(foreground)/BG(background) Detection Module, 2) Blob Entering Detection Module, 3) Blob Tracking Module, and 4) Trajectory Post Processing.

Suhr *et al.* [4] used a mixture of Gaussians (MoG) to model the background in a Bayer-pattern domain and classified the foreground in an interpolated red, green, and blue (RGB) domain. Hsieh *et al.* [5] used an adaptive background updating method to model the background. Kanhere and Birchfield [6] stored the average gray level of each pixel over a fixed period of time to train background model. del-Blanco *et al.* [7] used the mixture of Gaussians to model background and a Bayesian model to simulate the object trajectories in the tracking stage.

Except for detecting moving objects in the pixel domain, there were many studies during the last decade to detect moving objects based on analyzing a compressed domain video. Huang *et al.* [8] presented the quadtree structure to represent variable-size coding blocks in the high efficiency video coding (HEVC). This structure can represent the foreground objects in the picture. Nightingale *et al.* [9] proposed a heuristic, no-reference approach to classify video content which is specific to HEVC encoded bitstreams.

2169-3536 © 2016 IEEE. Translations and content mining are permitted for academic research only.
Personal use is also permitted, but republication/redistribution requires IEEE permission.
See http://www.ieee.org/publications_standards/publications/rights/index.html for more information.

Dey and Malay Kundu [10] proposed an efficient approach to extract foreground by using novel spatio-temporal decorrelated block features that was extracted directly from the HEVC compressed video.

Babu *et al.* [11] used the K-Means [12] clustering to estimate the number of moving objects, and then used the EM method to segment moving objects. Zeng *et al.* [13] proposed four types of motion vector: 1) background MV (BMV), 2) edge MV (EMV), 3) foreground MV (FMV), and 4) noise MV (NMV). He then used Markov Random Field (MRF) model to extract moving blocks. Poppe *et al.* [14] used a background video stream training background model, and then extracted foreground by background subtraction to get moving objects. Käs *et al.* [15] used Gaussian Mixture Model (GMM) to establish a background model with fully decoded I frame and detected the direction of moving objects in B, P frame. When foreground has been extracted by background subtraction, SIFT [16] was used to extract features and moving objects were tracked according to the features. Sabirin and Kim [17] used non-zero vectors to distinguish foreground and background, and then detected and tracked moving objects by the Spatio-Temporal Graph. When two moving objects were occluded, the moving objects can be tracked according to the moving direction and residue. Wang *et al.* [18], [19] present an approach to extract moving objects based on the INTRA frame from the H.264/AVC compressed domain video.

In order to improve the accuracy of moving object detection, some previous works tried to remove moving vectors that were caused by noises during the foreground detection. Yu *et al.* [20] used the median filter to remove noise. Ahmad *et al.* [21] used both the Gaussian filter and the median filter to remove noise. Ibrahim and Rao [22] combined temporal filter with spatial filter as an extended vector median filter to remove noise. Moura and Hemerly [23] proposed the concept of "fake movement", and used the spatiotemporal motion vector consistency filter (STF) to remove the fake movements. Kapotas and Skodras [24] used the mean value of motion vectors as the threshold to filter noise. Thus, to detect moving objects in compressed domain based on motion vectors becomes a feasible solution as seen from the above researches. The detection of moving objects and the estimation of numbers by using motion vectors of H.265/HEVC stream is proposed in this paper. The experimental environment used static cameras to monitor whether moving objects have crossed a preset tripwire or not. The moving objects are counted in the single camera view, cross-camera is not considered in our method. Wang *et al.* [25] proposed the motion vectors in B frames are much closer to the true motion direction of a moving object, and therefore this paper is focused on analyzing the motion vectors in B frames.

A simple block diagram of the proposed method is shown in Fig. 1. There are mainly four steps in the proposed method: 1) Foreground Detection, 2) Region of Interest (ROI) Detection, 3) Moving Object Tracking, and 4) Moving Object Counting.

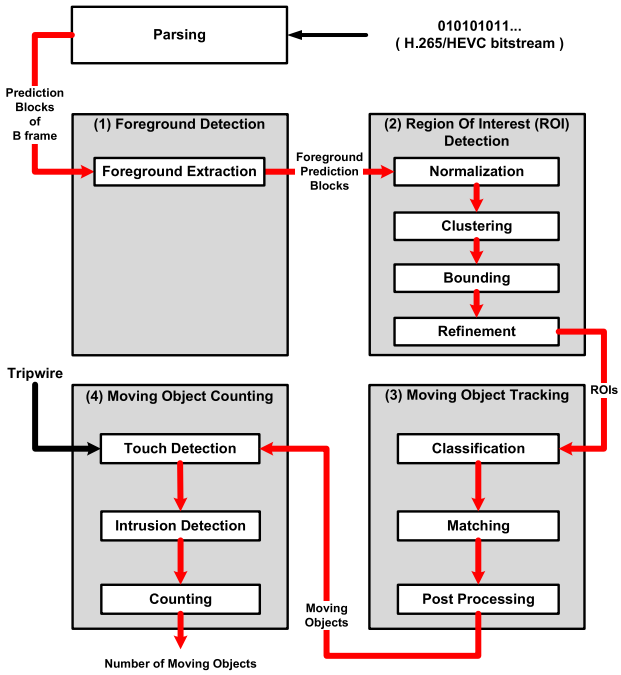


FIGURE 1. Block diagram of the proposed moving object counting.

The number of moving objects that move toward a predefined direction can be estimated through the above four steps. The result shows the estimated number of moving objects is very close to the real number. Thus, an intelligent video surveillance system can count moving objects by analyzing the motion vectors of B frames from H.265/HEVC bitstreams. Each step has its sub-steps and will be detailed in the following sections.

II. METHODOLOGY

A. MOTION VECTORS OF PREDICTION BLOCK

The basic encoding unit of H.265/HEVC is the Coding Tree Unit (CTU) [26]. Each image is divided into many CTUs. Every CTU is composed of an L × L luminance Coding Tree Block (CTB) and two L/2 × L/2 chrominance CTBs, as shown in Fig. 2.

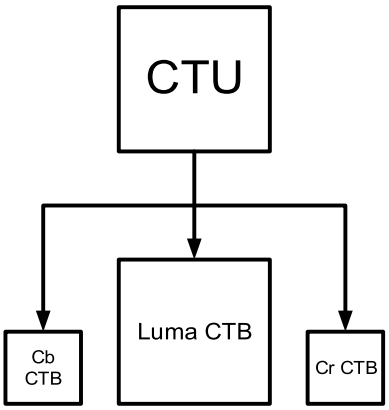


FIGURE 2. A CTU with a luma CTB and two chroma CTBs.

Size L is defined in SPS (Sequence Parameter Sets), it may be 16, 32 or 64. Every CTB will be divided into many Coding Blocks (CBs) according to its image feature; then used as a quarter tree to manage these CBs, as shown in Fig. 3.

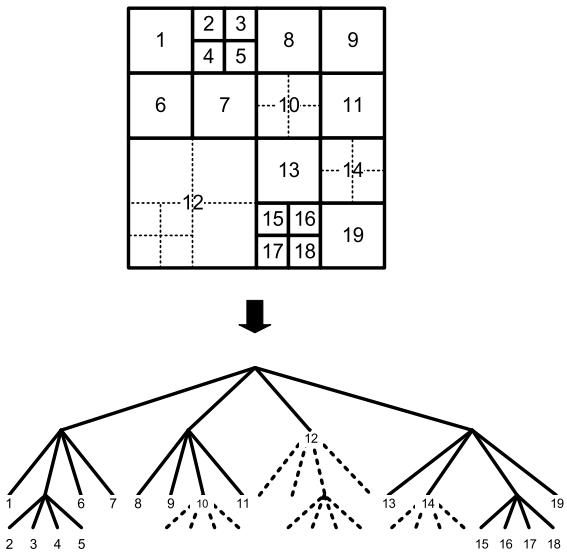


FIGURE 3. CTBs can be further partitioned into multiple CBs.

In general, the minimum size of a CB is 8 × 8, and is defined in SPS. Each CB is split into many Prediction Blocks (PBs) at the Inter Frame Prediction stage. PB sizes can be from 4 × 4 to 64 × 64, and there are two kinds of approach: symmetric or asymmetric, as shown in Fig. 4.

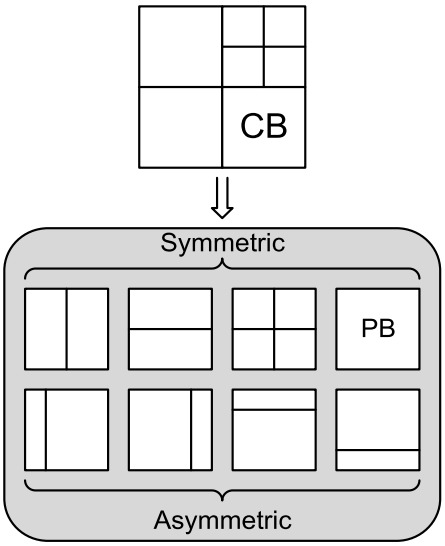


FIGURE 4. Splitting a CB into PBs.

When a CB is divided into many PBs, the encoder uses past and future frames to be the reference frames (in B frames, for example), and then finds the nearest matching block in reference frames by motion estimation. The motion vectors obtained by motion estimation are used in motion compensation prediction. In the following subsections, all the four steps in II are described in detail respectively.

B. FOREGROUND DETECTION

Foreground detection is used to extract the foreground prediction blocks. In this step, a novel momentum definition is proposed to reduce the impact of fake movements.

1) MOVEMENT OF PREDICTION BLOCK

Movement is the total displacement of a prediction block; it is calculated as below:

$$\begin{cases} Movement_i = \infty, & |R_i| = 0 \\ Movement_i = \|Mv_1\|_2\,, & |R_i| = 1 \\ Movement_i = \|\min Mv\|_2\,, & |R_i| = 2 \end{cases} \tag{1}$$

where

$\|R_i\|$:	Number of reference frames
Mv_1:	First motion vector of prediction block, $Mv_1 = (Mv_{1x}, Mv_{1y})$
Mv_2:	Second motion vector of prediction block, $Mv_2 = (Mv_{2x}, Mv_{2y})$
$\min Mv$:	$(\min(\|Mv_{1x}\|, \|Mv_{2x}\|), \min(\|Mv_{1y}\|, \|Mv_{2y}\|))$
i:	Prediction block index

Table 1 shows the movement properties of foreground and background prediction blocks. Foreground prediction blocks have high and wide variance movement because moving objects move at different speeds. In contrast, background prediction blocks have low and small variance movement because moving objects do not move, so that the movement value should be zero theoretically.

TABLE 1. Properties of movement.

	Foreground	*Background*
Value	High	Low
Variance	Wide	Narrow

2) FAKE MOVEMENT

The encoder will select the most similar block in another frame to be the reference block during motion estimation. The most similar block is not necessary at the same position even if the prediction block belongs to the background. Thus, background prediction blocks may have non-zero motion vectors that should be zero. This kind of movement with non-zero motion vector that should be zero is called "fake movement". In general, the encoder can find more easily a distant block from multiple similar reference blocks causing fake movement in the low frequency area or low-quality images.

3) MOMENTUM

Momentum is a weighted movement, and is a proposed novel concept. This new concept is used to identify fake movements more precisely. Momentum is calculated according to:

for the j-th CTB,

$$Momentum_i^{(j)} = Movement_i^{(j)} \times Mass_i^{(j)} \tag{2}$$

where

$Movement_i^{(j)}$:	Movement of i-th prediction block in j-th CTB
$Mass_i^{(j)}$:	Number of prediction blocks in j-th CTB. Thus, $Mass_1^{(j)} = Mass_2^{(j)} \ldots$ For example, if a CTB is divided into 7 prediction blocks,then every Mass of prediction block in the CTB is 7.
i :	Prediction block index
j :	CTB index

Foreground CTB is often divided into many prediction blocks so that foreground prediction blocks have higher $Mass_i^{(j)}$ than background. Foreground and background prediction blocks represent different distribution characteristics of $Movement_i^{(j)}$ and $Mass_i^{(j)}$, it is shown in Table 2.

TABLE 2. Distribution characteristics of momentum.

$Movement^{(j)}{}_i$	$Mass^{(j)}{}_i$	*Distribution Characteristics*
High	High	Foreground
Low	High	Foreground has low Movement
High	Low	Background has Fake Movement
Low	Low	Background

When $Movement_i^{(j)}$ and $Mass_i^{(j)}$ are both high, the prediction block most likely belongs to foreground. If $Movement_i^{(j)}$ and $Mass_i^{(j)}$ are both low, prediction block belongs to background in general. When the condition is low $Movement_i^{(j)}$ and higher $Mass_i^{(j)}$, the prediction block belongs to moving object at low speed. When the condition is high $Movement_i^{(j)}$ and low $Mass_i^{(j)}$, the prediction block may have fake movement.

4) FOREGROUND EXTRACTION

The set of all prediction blocks ($\mathbb{T}$) can be divided into two mutually exclusive sets:

A. Foreground prediction block set ($\mathbb{F}$)
B. Background prediction block set ($\mathbb{B}$)

In order to detect the moving objects, only set $\mathbb{F}$ need to be extracted with the next two steps:

Step 1. Calculate Threshold

The threshold is calculated according to:

$$MM = \frac{1}{|\mathbb{T}|} \sum_{Momentum \in \mathbb{T}} Momentum \tag{3}$$

where

$\mathbb{T}$:	Set of all prediction blocks ($\mathbb{T}$)
MM :	Mean *Momentum of* set of all prediction blocks ($\mathbb{T}$)

Step 2. Collect Foreground Prediction Blocks

The prediction blocks satisfying (4) are collected in the set $\mathbb{F}$.

$$\{PB_i | Momentum_i > MM\} \tag{4}$$

where

PB_i :	i-th prediction block
$Momentum_i$:	*Momentum* of i-th prediction block
i :	Prediction block index

The proposed momentum calculation can improve the difference between fake movement and others. A prediction block belonging to a moving object at low speed has a smaller movement. These kinds of prediction blocks are easily to be classified as background according to the movement only. Thus, movement multiplied by mass makes these prediction blocks to have high momentum. Thus, a moving object with low speed can be discriminated by its momentum. Similarly, a prediction block that has fake movement is easily misclassified as foreground by movement only. With the momentum calculation, these false-negative fake movements with very small momentum values will greatly reduce the false-negative decision probability.

C. REGION OF INTEREST (ROI) DETECTION

Region of Interest (ROI) is the possible area position of moving objects in the current frame. Each ROI is formed by merging foreground prediction blocks belonging to the same cluster. In this step, foreground prediction blocks will be clustered and then merged into a ROI. When a ROI has been detected, pseudo gravitation is proposed to refine it.

1) NORMALIZATION

The purpose of normalization is to convert prediction blocks with different sizes into blocks with the same size. According to the H.265/HEVC specification, the minimum prediction block has size 4×4. Therefore larger prediction blocks can be divided into 4×4 small blocks. Index Map is created to store these blocks. The Index Map is a matrix where each item value is 0 or 1 (default is 0) and the size is 1/16 of the original image (both the width and height are 1/4 of the original). Each pixel in the Index Map corresponds to a 4×4 block, as shown in Fig. 5.

The Index Map is used to fully express the distributions of prediction blocks of an image. Prediction blocks can be clustered on a fair basis because different sizes of prediction blocks are converted into the same size of Index Map pixels.

2) CLUSTERING

The clustering is done through applying the method in [27] to the Index Map. In [27], the connected-component labeling algorithm is introduced to find all the connected-components.

3) BOUNDING

Bounding is to create a rectangle that contains all the pixels in the same cluster. The rectangle is defined as two coordinates: start (startX, startY) at the Top-Left corner and end

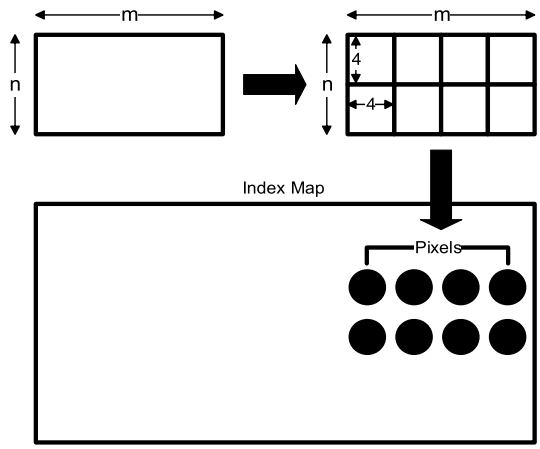


FIGURE 5. Mapping prediction blocks into pixels of Index Map.

(endX, endY) at the Bottom-Right corner. The rectangle is calculated according to:

$$\begin{cases} startX = \min(\mathbb{X}) \\ startY = \min(\mathbb{Y}) \\ endX = \max(\mathbb{X}) \\ endY = \max(\mathbb{Y}) \end{cases} \tag{5}$$

where

$\mathbb{S}$: $\{S_i | S_i \in \mathbb{R}^2, i = 1 \ldots n\}$
$\mathbb{X}$: $\{X_i | X_i$ is the first element of S_i in $\mathbb{S}\}$
$\mathbb{Y}$: $\{Y_i | Y_i$ is the second element of S_i in $\mathbb{S}\}$
i : Rectangle index

Coordinates of start (startX, startY) and end (endX, endY) will be multiplied by 4 to create a ROI after Bounding.

4) REFINEMENT

There are smaller ROIs caused by some foreground prediction blocks that are classified incorrectly as background prediction blocks because their momentum are below *MM*. These prediction blocks can cause fragmentation of foreground prediction blocks if their locations are not near the edge of the ROI. The squared distance of foreground prediction blocks belonging to the same ROI should be very close to each other. Therefore, these foreground prediction blocks need to be integrated to one ROI to reduce the unwanted ROI.

In order to get a more appropriate ROI, smaller ROIs are merged into a larger ROI attached to them. The distinction between a larger and a smaller ROI is according to:

$$MA = \frac{1}{n} \sum_{i=1}^{n} area_i \tag{6}$$

where

MA : Mean area of the ROIs
$area_i$: Area of i-th ROI_i
n : Number of ROIs
i : ROI index

Algorithm 1 Refinement Algorithm

```
FUNCTION Refinement (SmallerList, LargerList)
  FOR smallerId ← 0 to length(SmallerList)
    smaller ← SmallerList[smallerId]
    bestMatch ← smaller
    maximalForce ← 0

    FOR largerId ← 0 to length(LargerList)
      larger ← LargerList[largerId]
      f ← gravitation (larger, smaller)

      IF f > maximalForce then
          bestMatch ← larger
          maximalForce ← f
      ENDIF
    ENDFOR

    IF bestMatch ≠ smaller
      bestMatch ← merge(bestMatch, smaller)
      LargerList ← modify(LargerList, bestMatch)
    ELSE
      LargerList ← append(LargerList, smaller)
      ENDIF
    ENDFOR

  RETURN LargerList
ENDFUNCTION
```

An area greater than *MA* is defined as a larger ROI, while an area equal to or smaller than *MA* is a smaller ROI. Whether two ROIs are merged or not is determined by the pseudo gravitation force defined as:

$$\begin{cases} F = 0, & r \geq CTB\ Size \\ F = G\dfrac{m_1 m_2}{r^2}, & r < CTB\ Size \end{cases} \tag{7}$$

where

F : Pseudo gravitation force between two rectangles
G : The gravitational constant (always set to 1 in our approach)
m_1 : Area of first ROI
m_2 : Area of second ROI
r : The shortest boundary distance between two ROIs according to Fig. 6.

The pseudo gravitation is inversely proportion to the boundary distance between two ROIs. If r is larger than the size of CTB (64 in general), the pseudo gravitation is zero. A larger ROI will merge a smaller ROI according to the Refinement Algorithm shown in Algorithm 1.

The initial value of SmallerList is the ROIs whose area smaller than the MA, and the other ROIs whose area equal or bigger than the MA will be stored in the LargerList. In the Refinement Algorithm, a smaller ROI merges to a ROI with the maximum pseudo gravitation around it. If a smaller ROI cannot find any ROI to merge, it will be upgraded to become

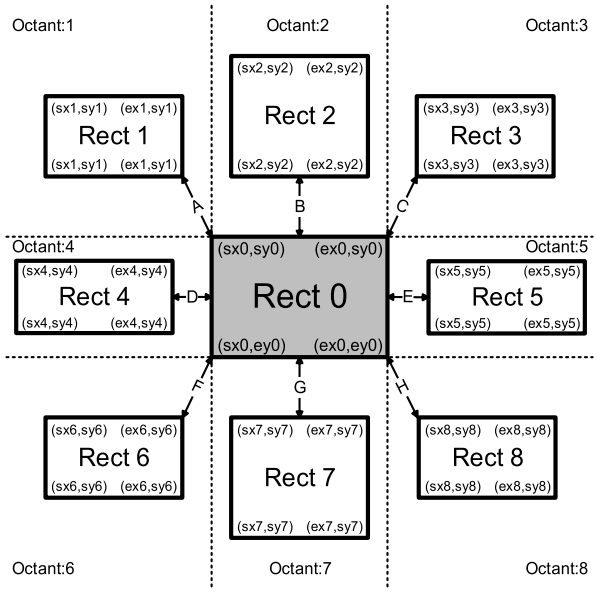


FIGURE 6. Shortest boundary distance between two rectangles. For example: The shortest boundary distance between Rect 0 and Rect 7 is G= |ey0-sy7|. The shortest boundary distance between Rect 0 and Rect 8 is $H = ((ex0\text{-}sx8)^2 + (ey0\text{-}sy8)^2)^{0.5}$.

a larger ROI. Eventually, the larger ROI will become a more appropriate ROI to represent the area position of moving object in the current frame.

D. MOVING OBJECT TRACKING

Moving Object Tracking is to identify the state of moving objects in the current frame. In this step, moving objects are tracked through matching moving objects and ROIs. When moving objects have been tracked, direction is calculated according its state.

1) CLASSIFICATION

The difference between ROI and moving object is identity. If a ROI is identified to a moving object that appear in the past frame, it will be the state of the moving object in the current frame. On the other hand, a ROI will be a new moving object if there is no past moving object matching to it. Classification is to identify the corresponding relations between ROIs and moving objects. These relations are established according to the following steps:

Step 1. Find Boundary Distance

Boundary distance between ROI and moving object is the shortest boundary distance between two rectangles according to Fig. 6.

Using of boundary distance between two moving objects can be more accuracy than centroid distance to determine whether two moving objects touch each other.

Step 2. Generate Grade Table

A moving object may have many ROIs to be its destinations and a ROI may be the destination of many moving objects. So each boundary distance between a ROI and a moving object needs to be given a Grade to identify the corresponding relation.

TABLE 3. Grade table.

	Moving Object₁	*Moving Object₂*	...
ROI₁	$G_{R_1O_1}, G_{O_1R_1}$	$G_{R_1O_2}, G_{O_2R_1}$	...
ROI₂	$G_{R_2O_1}, G_{O_1R_2}$	$G_{R_2O_2}, G_{O_2R_2}$	...
...	...	...	...

The number of rows (columns) of GRADE TABLE is the same as the number of ROIs (moving objects). Each entry pair in the Grade Table represents "Grade of a ROI to a moving object" and "Grade of a moving object to a ROI". An example pair is illustrated in Fig. 7.

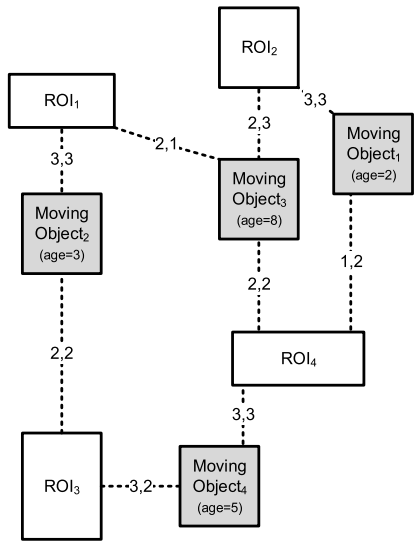


FIGURE 7. $(G_{R2O3}, G_{O3R2}) = (2, 3)$ means moving object$_3$ is the second close moving object to the ROI$_2$, thus giving grade 2 and ROI$_2$ is the most closest ROI to the moving object$_3$, thus giving grade 3.

Grade is defined as an integer from k to zero. The highest grade is k, the lowest grade is 1, and zero grade means no relation. If the Grade of relation from ROI$_i$ to moving object$_j$ is k, it means the moving object$_j$ is the closest moving object to the ROI$_i$. Grade is stored as a pair in the Grade table, as shown in Table 3.

All the Grades between ROIs and moving objects are included in the Grade Table.

Step 3. Calculate Matching Degree

Matching degree is the corresponding relation between ROIs and moving objects. The matching degree of each element in the Grade Table is calculated according to:

$$MatchingDegree_{ij} = G_{R_iO_j} \times G_{O_jR_i} \times A_j \tag{8}$$

where

G_{RiOj} : Grade of i-th ROI to j-th moving object
G_{OjRi} : Grade of j-th moving object to i-th ROI
A_j : Age of j-th moving object
i : ROI index
j : Moving object index

Matching degree is the product of "Grade of ROI to moving object", "Grade of moving object to ROI" and "Age of moving object". Age is the tracked times of a moving

object and used to represent the stability of a moving object. A moving object with larger age is a stable moving object that appears for a long period of time. On the other hand, a moving object with smaller age is an unstable moving object that appears for a short period of time.

Step 4. Create Degree List

A matching degree that is calculated in Step 3 is stored in the "DegreeList". Each element in the DegreeList contains three kinds of information: MatchingDegree, i-th ROI and j-th moving object as shown below:

$$[\text{MatchingDegree}_{ij}, \text{ROI}_i, \text{MovingObject}_j]$$

where

$\text{MatchingDegree}_{ij}$:	MatchingDegree between ROI_i and moving object_j
ROI_i :	i-th ROI in the RoiList
MovingObject_j :	j-th moving object in the ObjectList
RoiList :	All ROIs in the current frame
ObjectList :	All moving objects in the past frame
i :	ROI index
j :	Moving object index

The number of elements in the DegreeList is equal to the number of elements in the Grade Table.

2) MATCHING

Matching is to find the best corresponding relation between ROIs and moving objects in the DegreeList according to the Matching Algorithm as shown in Algorithm 2.

The proposed algorithm has three input lists: DegreeList, RoiList and ObjectList. It produces a outputs: ObjectList that stores moving objects in the "current frame". The initial entries of RoiList and ObjectList are all of the ROIs and moving objects.

The Matching Algorithm first sorts DegreeList by the MatchingDegree, from high to low. Secondly, it creates the MatchingList with empty initial value. Thirdly, it checks every element in the MatchingList. If the MatchingDegree between i-th ROI and j-th moving object is greater than 0, i-th ROI and j-th moving object are both still exist in the RoiList and ObjectList, then add i-th ROI and j-th moving object as a MatchingPair to the MatchingList. When a new MatchingPair is added to the Matching List, i-th ROI and j-th moving object are removed from the RoiList and ObjectList. An element in the MatchingList means "j-th moving object is tracked in the current frame, it is i-th ROI". The remainder ROIs in the RoiList means "i-th ROI is a moving object that first appears in the current frame". The remainder moving objects in the ObjectList means "j-th moving object vanished from the current frame". Finally, the vanished moving object will be deleted from the ObjectList, and the tracked moving object will update its attributes according to the ROI corresponding to it and the new moving object will be added in the ObjectList. A new moving object will be assigned a unique identity number to initialize. In our approach, maximum identity number of moving object in the ObjectList plus one will be the unique identity number that is assigned to the new moving object.

Algorithm 2 Matching Algorithm

```
FUNCTION Matching(DegreeList, RoiList, ObjectList)
  DegreeList ← QuickSort(DegreeList)
  MatchingList ← emptyList()

  FOR elementId ← 0 to length(DegreeList)
    element ← DegreeList[elementId]
    degree_ij ← element[0] //Get Degree
    roi_i ← element[1] // Get i-th ROI
    object_j ← element[2] // Get j-th Moving Object

    thereIsROI ← isExist(RoiList, roi_i)
    thereIsOBJECT ← isExist(ObjectList, object_j)

    IF degree_ij>0 AND thereIsROI AND thereIsOBJECT
      pair_ij ← [roi_i, object_j]
      MatchingList ← add(MatchingList, pair_ij)
      RoiList ← remove(RoiList, roi_i)
      ObjectList ← remove(ObjectList, object_j)
    ENDIF
  ENDFOR

  ObjectList ← deleteVanish(ObjectList)
  ObjectList ← update(ObjectList, MatchingList)
  ObjectList ← addAppear(ObjectList, RoiList)

  RETURN ObjectList
ENDFUNCTION
```

3) POST PROCESSING

a: SIZE CORRECTION

Even the same moving object may have different sizes in two adjacent frames, especially the moving object is composed of the prediction blocks near the edge of moving object. Therefore, the size of a moving object is often fluctuating. In order to stabilize it, a stable value S is calculated according to:

for the i-th moving object,

$$S_t^{(i)} = S_{t-1}^{(i)} - \frac{1}{p}F_{t-p}^{(i)} + \frac{1}{p}F_t^{(i)} \tag{9}$$

where

$S_t^{(i)}$:	Stable value in the current frame
$S_{t-1}^{(i)}$:	Stable value in the previous frame
t :	Current frame index
p :	Period
$F_{t-p}^{(i)}$:	Fluctuation value of moving object_i in frame_{t-p} (Fluctuation value is defined as $[\text{Size}^{(i)}]^T$)
$F_t^{(i)}$:	Fluctuation value of moving object_i in frame_t
$\text{Size}^{(i)}$:	Size of i-th moving object
i :	Moving object index.

Selection of the p will affect the value of S. When p is too short, it is insufficient to estimate S. When p is too long, it fails to estimate the actual size changes. So an appropriate p must be chosen for correct estimation. According to our observation, $p = 8$ can get a suitable S value. When S_t is calculated, the size of moving object is updated to S_t, and the center position of the moving object is calculated based on the size. Each re-calculation is stored and concatenated as an estimated trajectory.

b: DIRECTION FINDING

Direction of a moving object is the foundation of intrusion detection, because any intrusion object needs to satisfy its direction criterion within the range of intrusion direction.

Direction of moving object is a vector direction pointing to the moving object after a period of time. Period p is still used here and the direction is estimated according to:

$$\theta_O = \tan^{-1}(\frac{y}{x}) \tag{10}$$

where

θ_O	Direction of moving object
(x,y) :	*MovingVector* (Difference between the center of the moving object at the two different times)
MovingVector :	$center_t$ - $center_{t-period}$
$center_t$:	Coordinate of center of moving object in the current frame
$center_{t-period}$:	Coordinate of center of moving object in the past frame over a period of time.

E. MOVING OBJECT COUNTING

Moving Object Counting is to estimate the number of moving objects which move toward a pre-defined direction. In this step, tripwire is used to detect moving object intrusion according to its direction. Moving objects which intrude the tripwire for the first time will be counted.

1) TOUCH DETECTION

Tripwire is a finite-length line. The relation between moving object and tripwire is represented in Fig. 8.

D is the squared distance between the center of moving object and the tripwire. In order to detect whether a moving object touches the tripwire, (11) is used to determine when the condition is satisfied.

$$D < R \tag{11}$$

where

D :	$\frac{2\sqrt{S(S-a)(S-b)(S-c)}}{c}$
S :	$\frac{a+b+c}{2}$
a,b,c :	Three edges of triangle in Fig. 8.
R :	$\min(\frac{width_i}{2}, \frac{height_i}{2})$
$width_i$:	Width of i-th moving object
$height_i$:	Height of i-th moving object
i :	Moving object index

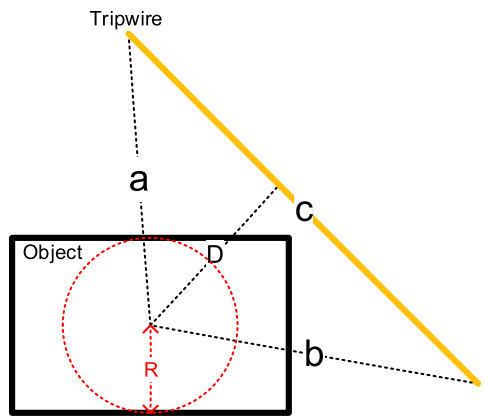


FIGURE 8. Relation between moving object and tripwire.

D is derived from [28]. When (11) is satisfied, the moving object touches the tripwire.

2) INTRUSION DETECTION

When a moving object touches the tripwire, it is checked whether (12) is satisfied.

$$\theta_T - \theta_O \in [\theta_l, \theta_u] \tag{12}$$

where

θ_T - θ_O :	Relative direction of moving object
θ_T :	Direction of tripwire
$[\theta_l, \theta_u]$:	Pre-defined range of relative intrusion direction

When both conditions (11) (12) are satisfied, the moving object is determined that it intrudes the tripwire.

3) COUNTING

When a moving object intrudes the tripwire, Equation (13) is used to determine whether the moving object should be added to the intrusion count.

$$A_i >= AL \tag{13}$$

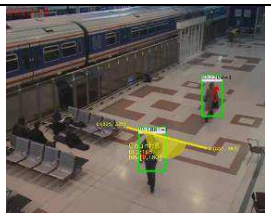
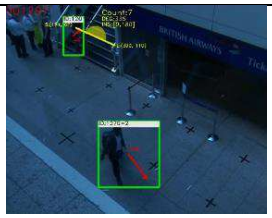
where

A_i :	Age of i-th moving object
AL :	Age limit
i :	Moving object index

AL is often defined as 3. The moving object can be counted when it is tracked at least 3 times. The AL condition can reduce the false-positives (counting a vanishing moving object) caused by transient noise. Moving object that is generated by transient noise has low age due to short-time appearance. The A_i of moving object can reduce error counting that is caused by transient noise. Finally, When conditions (11) (12) (13) are all satisfied, moving object is defined as an intrusion moving object. In order to avoid repeated counting, the tripwire will detect whether (14) is satisfied.

$$\begin{cases} Count + 1, & IMO_i \notin \mathbb{C} \\ Count, & otherwise \end{cases} \tag{14}$$

TABLE 4. Properties of test sequences.

	AVSS 2007 PV Hard	*PETS 2006 S1-T1-C 4*	*PETS 2007 S00 3*
Snapshot			
Frame Rate	25	25	25
Length	4110 (frames)	3022 (frames)	4500 (frames)
Resolution	720×576	720×576	720×576
Encoder	HM13	HM13	HM13
QP	32	32	32
Tripwire	Start:(622,477) → End:(464,549)	Start:(557,393) → End:(335,335)	Start:(194,62) → End:(300,110)
Scenario	Tripwire counts how many cars crossed it at the junction from the distant to the close.	Tripwire counts how many people crossed it from the distant to the close.	Tripwire counts how many people went upstairs.

where

$Count$:	Number of moving objects
IMO_i :	i-th moving object and it intrudes tripwire
$\mathbb{C}$:	$\{IMO_i \mid$ such that the properties of IMO_i satisfy (11)(12)(13)$\}$
i :	Moving object index

$\mathbb{C}$ is a set with empty initial value, and it is used to store moving object index that intrude tripwire after counting. The counter increases by 1 if the intrusion moving object is not already in $\mathbb{C}$; otherwise if the intrusion moving object has been counted before, it will not be counted again. Through the proposed method, the number of moving objects crossing the tripwire can be quickly estimated. This number is very close to the true number, therefore it is able to represent number of moving objects. The performance of our approach will be compared with the real number and another method in the next section.

III. EXPERIMENTAL RESULTS

There are three H.265/HEVC bitstreams test cases used as test sequences as shown in Table 4. The tripwire is placed in the path where moving objects cannot bypass the line inconsistent with the actual observation.

For the sake of comparison, we implement a pixel domain approach. The approach is also based on a five-stage process: Decoding, Foreground/Background Detection, Blob Entering Detection, Blob Tracking and Counting. The HM13 (High Efficiency Video Coding (HEVC) Encoder Description V13) [29] is used to implement Decoding stage. Foreground/Background Detection, Blob Entering Detection and Blob Tracking stages are implemented in terms of [30]–[32], they have already been a part of the standard library of OpenCV [33], it is called "blobtrack" [34]. The parameters of the function in OpenCV blobtrack are shown in [35]. The implementation of Counting stage is consistent with our approach. Because the first four stages that consume most of processing time are all implemented by standard library, so we believe that performance of the approach is effective.

There are two criteria of the evaluations: "Computation Performance" and "Counting Accuracy". "Computation Performance" shows the average execution time of each module in the proposed method and the pixel domain approach. "Counting Accuracy" shows the counting result every 500 frames.

A. COMPUTATION PERFORMANCE

Dell Precision T1700 Workstation is used as the test platform in our experiment. As opposed to the pixel domain approach, our approach is actually different from it in "Parsing", "Foreground Detection", "ROI Detection" and "Moving Object Tracking" stages. Hence, the average execution time of "Parsing", "Foreground Detection", "ROI Detection" and "Moving Object Tracking" of three test cases are shown in Table 5. These four stages are also called "Decoding", "FG/BG Detection", "Blob Entering Detection" and "Blob Tracking" in the pixel domain approach. The worst case execution time is by the test case AVSS 2007 AV Hard. Comparing to the pixel domain approach, our approach speed up to at least 400% (total execution time). This experimental result shows that our approach can greatly improve the performance of computation, particularly with large amount of video streaming data.

B. COUNTING ACCURACY

Massive works need to be done to get the true number of moving objects (ground truth) over the entire video stream. The counted moving objects are done 500 frames per comparison and the results are shown in Fig. 9.

In order to calculate the accuracy, there are four formulas to be used for evaluation: mean of square counting error (MSE),

TABLE 5. Moving object counting performance.

AVSS 2007 PV Hard						
	Parsing (Decoding)	*Foreground Detection (FG/BG Detection)*	*ROI Detection (Blob Entering Detection)*	*Moving Object Tracking (Blob Tracking)*	*Moving Object Counting*	*Total Time*
Our Approach	1.043 ms	18.643 ms	13.563 ms	2.177 ms	0.371 ms	35.797 ms
Pixel Domain Approach	5.622 ms	115.773 ms	0.862 ms	30.367 ms	0.008 ms	152.632 ms
PETS 2006 S1-T1-C 4						
	Parsing (Decoding)	*Foreground Detection (FG/BG Detection)*	*ROI Detection (Blob Entering Detection)*	*Moving Object Tracking (Blob Tracking)*	*Moving Object Counting*	*Total Time*
Our Approach	1.105 ms	11.477 ms	10.050 ms	1.305 ms	0.271 ms	24.208 ms
Pixel Domain Algorithm	4.225 ms	87.596 ms	0.890 ms	21.712 ms	0.007 ms	114.43 ms
PETS 2007 S00 3						
	Parsing (Decoding)	*Foreground Detection (FG/BG Detection)*	*ROI Detection (Blob Entering Detection)*	*Moving Object Tracking (Blob Tracking)*	*Moving Object Counting*	*Total Time*
Our Approach	1.117 ms	7.060 ms	8.309 ms	0.908 ms	0.225 ms	17.619 ms
Pixel Domain Algorithm	3.906 ms	88.613 ms	0.791 ms	21.277 ms	0.006 ms	114.593 ms

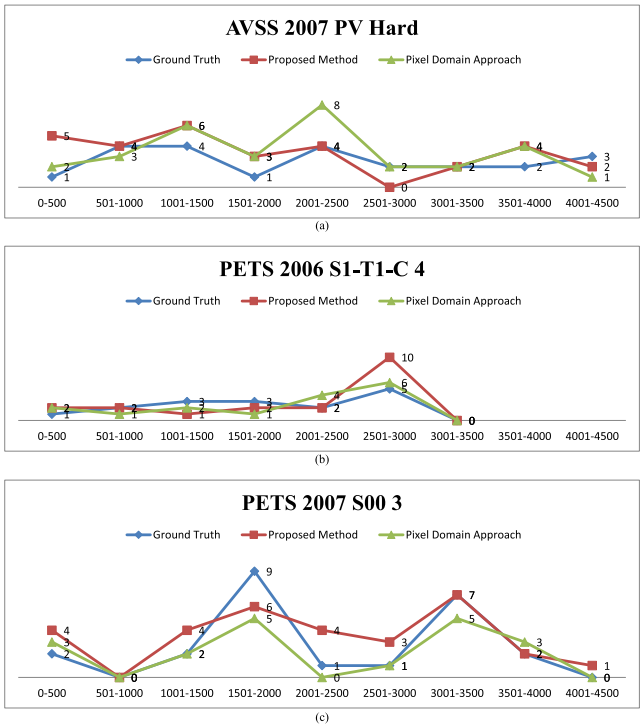


FIGURE 9. Variation between the counting number and time, the vertical axis is the number of counting, and the horizontal axis is 500 frame step. The blue line is ground truth number and the orange line is estimating number. (a) AVSS 2007 PV Hard, (b) PETS 2006 S1-T1-C 4 and (c) PETS 2007 S00 3.

mean of absolute counting error (MAE) [36], mean of missed counting error (MME), and mean of false alarm error (MFE). MSE and MAE are used to estimate accuracy of counting. Value of MSE and MAE are determined from (15) and (16), lower value means higher accuracy.

$$MSE = \frac{1}{n}\sum_{i=1}^{n}|c_i - g_i|^2 \tag{15}$$

where

n : Number of units (Every 500 frames is a unit, for example: 1000 frames are 2 units)
c_i : i-th counting result
g_i : i-th ground truth
i : Unit number

$$MAE = \frac{1}{n}\sum_{i=1}^{n}|c_i - g_i| \tag{16}$$

where

n : Number of units (Every 500 frames is a unit, for example: 1000 frames are 2 units)
c_i : i-th counting result
g_i : i-th ground truth
i : Unit number

Foreground prediction blocks belonging to two different clusters are too close to each other will be merged into a single ROI. When this kind of ROI is identified as an object, it will be counted only one time makes missed counting happen. MME is used to estimate missed counting error, which is False Negative. Value of MME is determined from (17), lower value means missed counting result occur infrequently.

$$MME = \frac{1}{n}\sum_{i=1}^{n}c_i^{miss} \tag{17}$$

where

n : Number of units (Every 500 frames is a unit, for example: 1000 frames are 2 units)
c_i^{miss} : $c_i^{miss} = \begin{cases} g_i - c_i, & g_i > c_i \\ 0, & otherwise \end{cases}$
c_i : i-th counting result
g_i : i-th ground truth
i : Unit number

TABLE 6. Moving object counting qualities.

	MSE	*MAE*	*MME*	*MFE*
AVSS 2007 PV Hard				
Our Approach	3.667	1.444	0.333	1.111
Pixel Domain Approach	3.778	1.555	0.333	1.222
PETS 2006 S1-T1-C 4				
	MSE	*MAE*	*MME*	*MFE*
Our Approach	4.429	1.285	0.429	0.857
Pixel Domain Approach	1.714	1.142	0.571	0.571
PETS 2007 S00 3				
	MSE	*MAE*	*MME*	*MFE*
Our Approach	3.444	1.444	0.333	1.0
Pixel Domain Approach	2.556	1	0.778	0.222

Camera-shaking, flash appear will cause some foreground prediction blocks that shouldn't be exist are extracted and merged into ROI. When this kind of ROI is identified as an object, false alarm can happen. MFE is used to estimate false alarm error, which is False Positive. Value of MFE is determined from (18), lower value means false alarm result occur infrequently.

$$MFE = \frac{1}{n}\sum_{i=1}^{n} c_i^{fa} \tag{18}$$

where

n : Number of units (Every 500 frames is a unit, for example: 1000 frames are 2 units)

c_i^{fa} : $c_i^{fa} = \begin{cases} c_i - g_i, & c_i > g_i \\ 0, & otherwise \end{cases}$

c_i : i-th counting result

g_i : i-th ground truth

i : Unit number

The results of evaluation are shown in Table 6. As Table 6 shows, MAE increases slightly by 0.111 in AVSS 2007 PV Hard. It can be reasonably inferred that the proposed method (red line) only increase 0.000222 (0.1/500) error per frame. It means increasing the error rate by 0.02%. As opposed to the pixel domain approach, our approach only increase the error rate by 0.02% and the speed up is 400%. This experimental result shows the proposed method can be widely applied to traffic management application.

IV. CONCLUSION

Automatically counting moving objects has played an important role in several different video surveillance applications. However, to obtain the estimation for a large volume of video data with high compression ratio is challenging due to the high computational cost. This paper presents a method for counting moving objects in the H.265/HEVC video format in the compressed domain.

In order to reduce the impact of fake movements, a new concept "Momentum" is proposed in this paper. The foreground prediction blocks can be extracted with minimum error by this concept. In the Region of Interest (ROI) Detection, connected-component labeling algorithm is used to cluster and pseudo gravitation is proposed to refine ROIs. In the Moving Object Tracking, state of moving object is determined in the current image frame according to the proposed Matching Algorithm. In the Moving Object Counting, the number of moving objects which move toward a specific direction is estimated by using tripwire. Besides finding the correct moving objects, an "Age" concept of moving object is proposed to reduce the influences caused by transient noises.

Based on the experimental results, the estimated number of moving objects is very close the true number. Compared with the pixel domain approach, our approach can increase the efficiency by 400% (from Table 5), while the average error per frame only increases by 0.02% (from Table 6). Hence, our research for traffic management based on H.265/HEVC bitstreams improves efficiency without damaging the counting quality too much.

REFERENCES

[1] *Information Technology—High efficiency coding and media delivery in heterogeneous environments—Part 2: High Efficiency Video Coding*, Standard ISO/IEC 23008-2:2013, ITU-T H.265, 2013.

[2] L. Anderson. (Feb. 2014). *H.265 Compression Set to Make a Mark on IP Video Surveillance*. [Online]. Available: http://us.sourcesecurity.com/news/articles/co-3130-ga-co-5188-ga.13056.html

[3] T. P. Chen *et al.*, "Computer vision workload analysis: Case study of video surveillance systems," *Intel Technol. J.*, vol. 9, no. 2, pp. 109–118, 2005.

[4] J. K. Suhr, H. G. Jung, G. Li, and J. Kim, "Mixture of Gaussians-based background subtraction for Bayer-pattern image sequences," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 21, no. 3, pp. 365–370, Mar. 2011.

[5] J.-W. Hsieh, S.-H. Yu, Y.-S. Chen, and W.-F. Hu, "Automatic traffic surveillance system for vehicle tracking and classification," *IEEE Trans. Intell. Transp. Syst.*, vol. 7, no. 2, pp. 175–187, Jun. 2006.

[6] N. K. Kanhere and S. T. Birchfield, "Real-time incremental segmentation and tracking of vehicles at low camera angles using stable features," *IEEE Trans. Intell. Transp. Syst.*, vol. 9, no. 1, pp. 148–160, Mar. 2008.

[7] C. R. del-Blanco, F. Jaureguizar, and N. García, "An efficient multiple object detection and tracking framework for automatic counting and video surveillance applications," *IEEE Trans. Consum. Electron.*, vol. 58, no. 3, pp. 857–862, Aug. 2012.

[8] T. Huang, S. Dong, and Y. Tian, "Representing visual objects in HEVC coding loop," *IEEE J. Emerg. Sel. Topics Circuits Syst.*, vol. 4, no. 1, pp. 5–16, Mar. 2014.

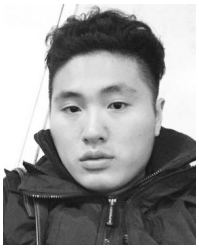
[9] J. Nightingale, Q. Wang, C. Grecos, and S. R. Goma, "Deriving video content type from HEVC bitstream semantics," *Proc. SPIE*, vol. 9139, pp. 913902-1–913902-13, May 2014.

[10] B. Dey and K. Malay Kundu, "Efficient foreground extraction from HEVC compressed video for application to real-time analysis of surveillance 'big' data," *IEEE Trans. Image Process.*, vol. 24, no. 11, pp. 3574–3585, Nov. 2015.

[11] R. V. Babu, K. R. Ramakrishnan, and S. H. Srinivasan, "Video object segmentation: A compressed domain approach," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 14, no. 4, pp. 462–474, Apr. 2004.

[12] J. MacQueen, "Some methods for classification and analysis of multivariate observations," in *Proc. 5th Berkeley Symp. Math. Statist. Probab.*, 1967, pp. 281–297.
[13] W. Zeng, J. Du, W. Gao, and Q. Huang, "Robust moving object segmentation on H.264/AVC compressed video using the block-based MRF model," *Real-Time Imag.*, vol. 11, no. 4, pp. 290–299, 2005.
[14] C. Poppe, S. De Bruyne, T. Paridaens, P. Lambert, and R. Van de Walle, "Moving object detection in the H.264/AVC compressed domain for video surveillance applications," *J. Vis. Commun. Image Represent.*, vol. 20, no. 6, pp. 428–437, 2009.
[15] C. Käs, M. Brulin, H. Nicolas, and C. Maillet, "Compressed domain aided analysis of traffic surveillance videos," in *Proc. 3rd ACM/IEEE Int. Conf. Distrib. Smart Cameras (ICDSC)*, Como, Italy, Aug./Sep. 2009, pp. 1–8.
[16] D. G. Lowe, "Distinctive image features from scale-invariant keypoints," *Int. J. Comput. Vis.*, vol. 60, no. 2, pp. 91–110, Nov. 2004.
[17] H. Sabirin and M. Kim, "Moving object detection and tracking using a spatio-temporal graph in H.264/AVC bitstreams for video surveillance," *IEEE Trans. Multimedia*, vol. 14, no. 3, pp. 657–668, Jun. 2012.
[18] F.-P. Wang, W.-H. Chung, and S.-Y. Kuo, "An efficient approach to extract moving objects by the H.264 compressed-domain features," in *Proc. 12th Int. Conf. ITS Telecommun.*, Taipei, Taiwan, Nov. 2012, pp. 452–456.
[19] F.-P. Wang, W.-H. Chung, G.-K. Ni, I.-Y. Chen, and S.-Y. Kuo, "Moving object extraction using compressed domain features of H.264 INTRA frames," in *Proc. IEEE 9th Int. Conf. Adv. Video Signal-Based Surveill.*, Beijing, China, Sep. 2012, pp. 258–263.
[20] X.-D. Yu, L.-Y. Duan, and Q. Tian, "Robust moving video object segmentation in the MPEG compressed domain," in *Proc. Int. Conf. Image Process. (ICIP)*, Barcelona, Spain, Sep. 2003, pp. III-933–III-936.
[21] M. A. A. Ahmad, D.-Y. Chen, and S.-Y. Lee, "Robust object detection using cascade filter in MPEG videos," in *Proc. IEEE 5th Int. Symp. Multimedia Softw. Eng. (ISMSE)*, Taichung, Taiwan, Dec. 2003, pp. 196–203.
[22] M. M. Ibrahim and S. Rao, "Motion analysis in compressed video—An hybrid approach," in *Proc. IEEE Workshop Motion Video Comput. (WMVC)*, Austin, TX, USA, Feb. 2007, p. 17.
[23] R. C. Moura and E. M. Hemerly, "A spatiotemporal motion-vector filter for object tracking on compressed video," in *Proc. 7th IEEE Int. Conf. Adv. Video Signal Based Surveill.*, Boston, MA, USA, Aug./Sep. 2010, pp. 427–434.
[24] S. K. Kapotas and A. N. Skodras, "Moving object detection in the H.264 compressed domain," in *Proc. IEEE Int. Conf. Imag. Syst. Techn. (IST)*, Thessaloniki, Greece, Jul. 2010, pp. 325–328.
[25] R. Wang, H.-J. Zhang, and Y.-Q. Zhang, "A confidence measure based moving object extraction system built for compressed domain," in *Proc. IEEE Int. Symp. Circuits Syst. (ISCAS)*, Geneva, Switzerland, May 2000, pp. 21–24.
[26] G. J. Sullivan, J.-R. Ohm, W.-J. Han, and T. Wiegand, "Overview of the high efficiency video coding (HEVC) standard," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 22, no. 12, pp. 1649–1668, Dec. 2012.
[27] R. C. Gonzalez and R. E. Woods, "2.5.2 adjacency, connectivity, regions, and boundaries," in *Digital Image Processing*, 2nd ed. Upper Saddle River, NJ, USA: Prentice-Hall, Jan. 2002, pp. 66–68.
[28] K. K. Johnson. (Jul. 2006). *Heron, Brahmagupta, Pythagoras, and the Law of Cosines Expository Paper*. [Online]. Available: http://scimath.unl.edu/MIM/files/MATExamFiles/JohnsonK_MAT_Exam_ExpositoryPaper.pdf
[29] *High Efficiency Video Coding (HEVC) Test Model 13 (HM13) Encoder Description*, document JCTVC-O1002, Nov. 2013.
[30] L. Li, W. Huang, I. Y. H. Gu, and Q. Tian, "Foreground object detection from videos containing complex background," in *Proc. 11th ACM Int. Conf. Multimedia*, Berkeley, CA, USA, Nov. 2003, pp. 2–10.
[31] D. da Silva Pires, R. M. Cesar, M. B. Vieira, and L. Velho, "Tracking and matching connected components from 3D video," in *Proc. 18th Brazilian Symp. Comput. Graph. Image Process. (SIBGRAPI)*, Oct. 2005, pp. 257–264.
[32] D. Comaniciu and P. Meer, "Mean shift analysis and applications," in *Proc. 7th IEEE Int. Conf. Comput. Vis.*, vol. 2. Sep. 1999, pp. 1197–1203.
[33] OpenCV. *Open Source Computer Vision Library*, accessed on Jun. 2015. [Online]. Available: http://opencv.org/
[34] Y.-W. Chen. *Blob Tracking Modules*, accessed on Jun. 2015. [Online]. Available: http://homepage.ntu.edu.tw/~d00921022/Blob_Tracking_Modules.htm
[35] Y.-W. Chen. *Blob Tracking Parameters*, accessed on Jun. 2015. [Online]. Available: http://homepage.ntu.edu.tw/~d00921022/Blob_Tracking_Parameters.htm
[36] R. Zhao and X. Wang, "Counting vehicles from semantic regions," *IEEE Trans. Intell. Transp. Syst.*, vol. 14, no. 2, pp. 1016–1022, Jun. 2013.

YUNG-WEI CHEN received the M.E. degree from the Department of Electrical Engineering, National Chung Cheng University, Chiayi, Taiwan, in 2007. He is currently pursuing the Ph.D. degree with the Department of Electrical Engineering, National Taiwan University, Taipei, Taiwan. From 2007 to 2009, after his graduation, he worked in security and surveillance industry. His research interests include intelligent video surveillance and video coding.



KAI CHEN received the master's degree in computer science from National Taiwan University, Taipei, Taiwan, in 2008, where he is currently pursuing the Ph.D. degree with the Department of Electrical Engineering. From 2008 to 2011, he joined Live-Streaming Team as a Senior Engineer with AVerMedia Inc., Taipei, worked for ATV, DTV, and Internet TV Project. His research interests include multimedia, video codec, and media streaming.

SHIH-YI YUAN received the Ph.D. degree in electrical engineering from National Taiwan University, in 1997. He has served as a Technical committee Member in several IEEE-sponsored conferences for past ten years and conducted several government projects with the Bureau of Standards, Metrology and Inspection, Ministry of Economic Affairs, China. He is currently an Associate Professor with the Department of Communication Engineering and the Section Chief of Planning of IC-EMC Center with Feng Chia University. His research interests include signal process, embedded software design, and software solution for EM-aware compiler design.

SY-YEN KUO (S'85–M'88–SM'98–F'01) received the B.S. degree in electrical engineering from National Taiwan University (NTU), Taipei, Taiwan, in 1979, the M.S. degree in electrical and computer engineering from the University of California at Santa Barbara, in 1982, and the Ph.D. degree in computer science from the University of Illinois at Urbana–Champaign, in 1987. He is the Dean of the College of Electrical Engineering and Computer Science and a Distinguished Professor with the Department of Electrical Engineering, NTU, and was the Chairman at the same department from 2001 to 2004. He took a leave from NTU and served as a Chair Professor and the Dean of the College of Electrical Engineering and Computer Science with the National Taiwan University of Science and Technology from 2006 to 2009. He spent his sabbatical years as a Visiting Professor with the Department of Computing, Hong Kong Polytechnic University, from 2011 to 2012, and the Computer Science and Engineering Department, The Chinese University of Hong Kong, from 2004 to 2005, and as a Visiting Researcher with AT&T Labs-Research, NJ, from 1999 to 2000, respectively. He was the Chairman of the Department of Computer Science and Information Engineering with National Dong Hwa University, Taiwan from 1995 to 1998, a Faculty Member with the Department of Electrical and Computer Engineering, University of Arizona, from 1988 to 1991, and an Engineer with Fairchild Semiconductor, CA, and Silvar-Lisco, CA, from 1982 to 1984. In 1989, he also worked as a Summer Faculty Fellow with the Jet Propulsion Laboratory, California Institute of Technology. His current research interests include dependable systems and networks, mobile computing, cloud computing, and quantum computing and communications. He has published more than 390 papers in journals and conferences, and also holds 13 U.S. patents, nine Taiwan patents, and several other patents. He received the Distinguished Research Award three times consecutively and the Distinguished Research Fellow from the National Science Council, Taiwan. He was also a recipient of the best paper award in the 1996 International Symposium on Software Reliability Engineering, the best paper award in the simulation and test category at the 1986 IEEE/ACM Design Automation Conference, the National Science Foundation's Research Initiation Award in 1989, and the IEEE/ACM Design Automation Scholarship in 1990 and 1991.