# Compressed-Domain Highway Vehicle Counting by Spatial and Temporal Regression

Zilei Wang, *Member, IEEE*, Xu Liu, Jiashi Feng, Jian Yang, *Senior Member, IEEE*, and Hongsheng Xi

***Abstract*— Counting on-road vehicles in the highway is fundamental for intelligent transportation management. This paper presents the first highway vehicle counting method in compressed domain, aiming at achieving comparable estimation performance with the pixel-domain methods. Counting in compressed domain is rather challenging due to limited information about vehicles and large variance in vehicle numbers. To address this problem, we develop new low-level features to mitigate the challenge from insufficient information in compressed videos. The new proposed features can be easily extracted from the coding-related metadata. Then, we propose a hierarchical classification-based regression (HCR) model to estimate the number of vehicles from the compressed-domain low-level features for individual frame. HCR hierarchically divides the traffic scenes into different cases according to the density of vehicles such that the large variance of traffic scenes can be effectively captured. Beside the spatial regression in each frame, we propose a locally temporal regression model to further refine the counting results, which exploits the continuous variation characteristics of the traffic flow. We extensively evaluate the proposed method on real highway surveillance videos. The experimental results consistently show that the proposed method is very competitive compared with the pixel-domain methods, which can reach similar performance with much lower computational cost.**

***Index Terms*— Vehicle counting, hierarchical classification, regression, compressed domain.**

## I. Introduction

ESTIMATING the number of on-road vehicles is one of the most important tasks in intelligent transportation system (ITS), which can be used to monitor the traffic status and provide information for traffic control and optimization, *e.g.*, better driving routes through bypassing the congested roads [1]. In this paper, we consider to solve the vehicle counting problem by developing the video analysis based methods. Compared with the systems employing specialized sensors (*e.g.* infrared or inductive loop detector), the visual counting system is easier to deploy by reusing the roadside cameras, and thus much less costly [2].

In a typical traffic surveillance system, the central subsystem connects all terminal cameras through a private network and receives the surveillance videos. In practice, however, only a part of video streams can be simultaneously accessed with variable bit-rate due to the bandwidth limitation [3]. Though acquiring such partial information is sufficient for monitoring with free stream switch, it is insufficient for semantic video analysis that needs all of the video streams for fully capturing the traffic status. Therefore, one usually conducts the analysis in terminal devices (*e.g.*, surveillance workstations). To reduce the overall cost, these devices are usually equipped with cheaper systems than central servers. Therefore, the video analysis algorithms are expected to have low computational complexity for fulfilling the computation resource limitation on these cheaper systems.

Video analysis algorithms can be implemented in either the pixel or the compressed domain. Particularly, the pixel-domain algorithms first decode the compressed surveillance videos into raw frames and then operate on the frame pixels. Due to involving such decoding procedure and massive pixels to process, the computational complexity is usually tremendously high [4]. In contrast, the compressed-domain video analysis algorithms are able to directly operate on the compressed video [5], and are far more less expensive. Hence the compressed-domain algorithms are more appropriate for real-time large-scale video analysis. Therefore, we propose to address the vehicle counting problem in the compressed domain.

The video analysis in compressed domain mainly relies on the encoding metadata, which can be easily extracted from video bitstreams, *e.g.* the motion vector (MV), DCT coefficients, and macro-blocks (MB) partition modes [5]. However, there are two challenges for vehicle counting in compressed domain. First, the critical metadata in compressed videos (*i.e.* motion estimation and compensation vectors) are designed for compression efficiency rather than video analysis [6]. Consequently, the features extracted from video bitstreams are noisy and cannot accurately describe moving vehicles compared with the raw frames. Second, the traffic scenes usually change fast, present varying numbers of vehicles, and vulnerable to the external factors (*e.g.* weather conditions, and illumination changes) [7]. These challenges make it quite difficult to accurately model the realistic traffic scenes for vehicle counting.

Manuscript received April 22, 2017; revised August 29, 2017; accepted October 2, 2017. Date of publication October 11, 2017; date of current version January 7, 2019. This work was supported in part by the National Natural Science Foundation of China under Grant 61673362 and Grant 61233003, in part by Youth Innovation Promotion Association CAS, and in part by the Fundamental Research Funds for the Central Universities. This paper was recommended by Associate Editor Y. Wang. *(Corresponding author: Zilei Wang.)*

Z. Wang, X. Liu, J. Yang, and H. Xi are with the School of Information Science and Technology, University of Science and Technology of China, Hefei 230027, China (e-mail: zlwang@ustc.edu.cn).

J. Feng is with the Department of Electrical and Computer Engineering, National University of Singapore, Singapore 117576.

Color versions of one or more of the figures in this paper are available online at http://ieeexplore.ieee.org.

Digital Object Identifier 10.1109/TCSVT.2017.2761992

In this paper, we propose a multi-regression method for highway vehicle counting in the compressed domain, aiming at achieving comparable prediction performance with the pixel-domain methods. To our best knowledge, this is the first research attempt on this challenging problem. Specifically, we first develop new low-level features to capture the crucial information useful for counting vehicles. These features can be easily computed from the provided MVs and block partition modes, and cover the size, shape, motion, and texture information of traffic scenes. Combining these features can effectively mitigate the challenges of information insufficiency for counting vehicles in the compressed domain. Then we propose a Hierarchical Classification based Regression (HCR) model for counting vehicles in a video frame. As illustrated in Figure 1, HCR divides the traffic scenes into multiple cases according to the vehicle density (*e.g.* heavy, medium, and light). Then HCR adopts the most suitable regression model for each individual case. Indeed, introducing such division can better approximate the large variance over characteristics of traffic scenes. Furthermore, we add one more classification layer for handling the ambiguity cases produced by the first-layer classifiers. As a result, the large estimation deviation incurred by misclassification can be greatly alleviated. Besides the spatial regression, we further propose a locally temporal regression method to refine the per-frame counting results, which exploits the continuous characteristics of the traffic flow. By combining the spatial and temporal regression, our proposed method can produce robust and accurate vehicle counting results.

We evaluate the proposed method on the real highway surveillance videos captured under various traffic scenes. The experimental results show that our method is very competitive with the conventional pixel-domain methods. Our method gives the similar performance while consumes much lower computational resource. This paper is an extended version of our previous work that appeared in [8]. It differs from [8] in the following aspects:

(1) We propose a locally temporal regression method to refine the estimated vehicle counts, and consequently the vehicle counting performance can be further improved.
(2) We propose a new compressed-domain LBP feature, which can be computed faster than the previous version in [8] without performance drop.
(3) We introduce a new vehicle counting dataset, US101 Highway, which more comprehensively justifies the effectiveness of the proposed method.
(4) We provide more details of the method and conduct more experiments.

The remainder of the paper is organized as follows. Section II discusses the related works. Section III elaborates on the proposed vehicle counting system, and Section IV experimentally evaluates the proposed method. Finally, Section V concludes this work.

## II. Related Works

Object counting is one of the classic visual recognition tasks that aims to estimate the number of specific objects within an image [9]. According to the adopted strategy, existing object counting approaches can be divided into three categories: counting by detection, counting by clustering, and counting by regression [10]. Among them, the most intuitive approach is the detection based one [11], [12]. The DPM model [13], which builds on HOG features [14], was a common choice for object detection [15]–[17]. More recent works [18]–[20] have resorted to deep convolutional neural network (DCNN), which has remarkably boosted the performance. However, the methods by detection [21] and clustering [22], [23] need to explicitly segment the objects or track the feature points, and may fail in presence of heavy serious occlusions or scene clutters. In contrast, the regression based methods [24]–[26] directly learn a mapping from the extracted features to density values. Such a strategy can effectively alleviate the negative effects of unavoidable interference and is more preferred in practice. Hence the regression model is usually considered more applicable to object counting for the realistic scenes.

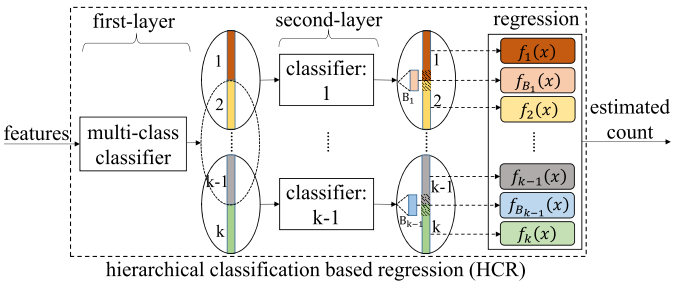



Fig. 1. The Hierarchical Classification based Regression (HCR) model. The traffic scenes are firstly classified into $k$ categories representing different vehicle densities. To tackle the fuzzy cases produced by the first-layer classifier, the second layer classifiers are introduced to distinguish the samples located in the boundary area (*i.e.* $B_1$ to $B_{k-1}$). Afterwards, the regression models are used to each category.

The regression methods have made significant progress for object counting in images. Davies *et al.* [27] proposed a linear regression model to map the holistic features to the people counting results. Chan *et al.* [24] proposed a perspective normalization method to handle the diversity of camera perspectives along with a bank of complementary features. To tackle the visual diversity of vehicle appearance (*e.g.* truck vs. car), a three-level cascaded regression model was proposed in [2]. In addition, some semi-supervised counting methods [28], [29] were also developed, which exploit the continuity and consistency between unlabeled samples and their temporally adjacent samples. These methods can exploit more unlabeled data, *e.g.* via transfer learning [30], and thus perform better for complex crowd scenes that are difficult to annotate. Recently, convolutional neural network (CNN) is introduced for crowd counting. For example, Zhang *et al.* [26] proposed a cross-scene counting model, which is fine-tuned upon a pretrained model for a new scene by feeding the retrieved similar training samples. In order to mitigate the negative influence of perspective distortion, Zhang *et al.* [31] proposed to employ filters of different sizes to handle large variation of people/head sizes. Onoro-Rubio and López-Sastre [32] proposed the Hydra CNN to learn a multi-scale nonlinear regression model with a pyramid of image patches at multiple

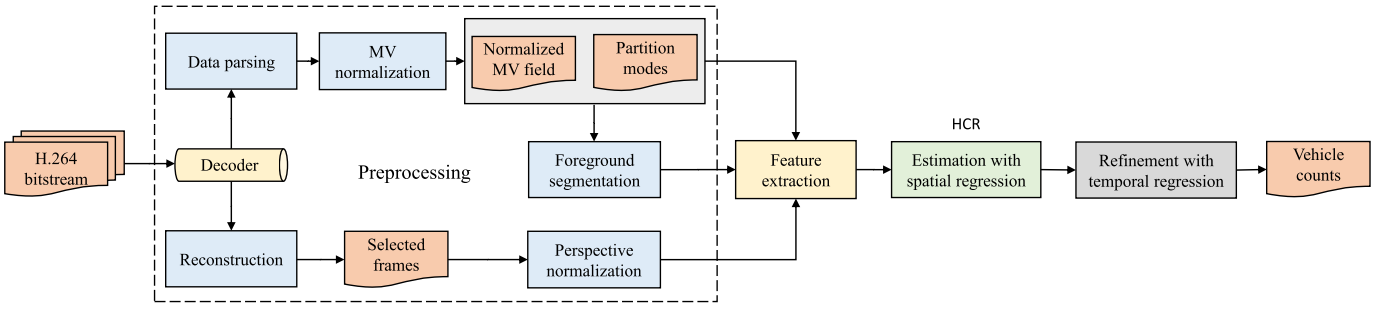



Fig. 2. The framework of the proposed highway vehicle counting system, which includes four key stages: (1) video preprocessing, (2) feature extraction, (3) estimation with spatial regression, and (4) refinement with temporal regression. The preprocessing stage is to extract the encoding information from input video bitstream, and then prepares the necessary data for feature extraction. Then these data are translated into various features to represent complex traffic scenes. Next, the number of vehicles in a frame is estimated using the proposed HCR. Finally, the temporal regression method is applied to refine the estimation results.

scales. Another remarkable work was proposed in [33], where multi-scale features from the pre-trained ResNets [34] are combined to estimate vehicle density.

However, to our best knowledge, no compressed-domain methods have been investigated for object counting. Actually, most works in compressed domain focus on the detection and segmentation of moving object [35], [36], object tracking [4], [37], face detection [38], and crowd flow segmentation [39]. Some related works mainly focus on estimating the traffic parameters, *e.g.*, the congestion level and vehicle speed. Specifically, Porikli and Li [40] proposed a traffic congestion estimation method by analyzing the MPEG-encoded videos, where the DCT coefficients and MVs are exploited. Tusch *et al.* [41] introduced four features related to the vehicle density to estimate level of service (LOS). Yu *et al.* [42] proposed to estimate the mean vehicle speed using the MVs of MPEG-encoded videos. Obviously, these works are different from the problem we will addressed in this work.

## III. Our Approach

### A. Overview

The main challenges to highway vehicle counting in compressed domain lie in the limited and inaccurate information and large variance of traffic scenes. In this work, we tackle these challenges by constructing rich low-level features, which can effectively exploit the available data and fully represent the traffic scenes. We also propose a novel counting model, *i.e.*, the Hierarchical Classification based Regression (HCR), which applies a suitable submodel for the given traffic scene in an adaptive manner to its presenting characteristics.

Figure 2 shows the framework of the proposed counting method. It contains the following four stages: (1) video preprocessing, (2) feature extraction, (3) estimation with spatial regression, and (4) refinement with temporal regression. Specifically, in the preprocessing stage, we first parse the input video bitstream to extract the video encoding information, and then prepare the necessary data for following feature extraction. Then we translate these data into various features that can accurately represent complex traffic scenes. Next we estimate the number of vehicles in each frame using the proposed HCR method. Finally, we refine the counting results by applying the proposed temporal regression method. In this paper, H.264 codec [6] is adopted considering its high encoding efficiency and wide application in the real video surveillance systems. For a given compressed video bitstream, we mainly extract the metadata of Motion Vector (MV) and Macro-block (MB) partition modes.

### B. Preprocessing

The preprocessing stage targets to produce the metadata of standardized format from the raw video bitstream, which include the motion vector normalization, macro-block weighting, foreground segmentation, and perspective normalization.

*1) Motion Vector Normalization:* In the H.264 compressed format, MB is the basic unit of video encoding [6]. The MBs can be encoded in various block partition modes, such as $16 \times 16$, $16 \times 8$, $8 \times 16$, $8 \times 8$, $8 \times 4$, $4 \times 8$, and $4 \times 4$, and each block corresponds to an MV. In addition, multiple reference frames can be used for one frame in order to improve the efficiency of inter-frame encoding. Thus the MVs in the same frame often have different temporal scales. In this work, we use the temporal interpolation to normalize the MVs to a uniform temporal scale. Let $B_{ij}$ denote the MB at the location $(i, j)$, where $i$ and $j$ denote the index of MB along the X-axis and Y-axis in the video frames, respectively. The MV of $B_{ij}$ at the time $t$ is denoted by $V_{ij}(t)$. The corresponding normalized MV is defined as:

$$\tilde{V}_{ij}(t) = \frac{V_{ij}(t)}{t - r}, \tag{1}$$

where $r$ is the time of the reference MB.

The mode of the smallest block in H.264 is $4 \times 4$. In order to obtain a uniform MV field, we split all the blocks into $4 \times 4$, *e.g.* one $8 \times 16$ would be partitioned into 8 pseudo-MBs with the size of $4 \times 4$. Particularly, the MVs of $4 \times 4$ pseudo-MBs are straightforwardly assigned using the MV of corresponding parent MB. Additionally, for the intra-coded blocks that have no MVs, we adopt the Polar Vector Median (PVM) [4] method to compute their MVs, *i.e.*, the vectors are computed in the polar coordinates. Formally, let $P = (\mathbf{v}_1, \mathbf{v}_2, \cdots, \mathbf{v}_n)$ be the list of MVs collected from the neighboring $4 \times 4$ blocks of the current intra-coded MB, where the elements are sorted

according to their angles from $-\pi$ to $+\pi$. We identify the index $\bar{i}$ with $m = (n+1)/2$ as

$$\bar{i} = \arg\min_{j} \sum_{i=j}^{j+m-2} \theta_i, \tag{2}$$

where $\theta_i$ denotes the angle between the vectors $\mathbf{v}_i$ and $\mathbf{v}_{i+1}$ (let $\mathbf{v}_1 \equiv \mathbf{v}_{n+1}$ here). Then we construct a collection of vectors as $P_s = (\mathbf{v}_{\bar{i}}, \mathbf{v}_{\bar{i}+1}, \cdots, \mathbf{v}_{\bar{i}+m-1})$, which actually contains approximately half of the original vectors in $P$ such that the sum of the angles between them is the minimal. Finally, we generate the MV of the intra-coded block vector as follows: the angle is set as the median of angles of the vectors in $P_s$, while its magnitude is set as the median of magnitudes of the vectors in $P$. In this way, we obtain a normalized MV field $\tilde{V}$ for each frame.

*2) Macro-Block Weighting:* There are seven different partition sizes in H.264 with the application of deformable macroblock technique. Particularly, the areas around moving objects usually have small partition sizes for higher compression efficiency [37]. Therefore, the blocks with smaller partition sizes are more likely to represent the actual vehicle motion. We assign MBs different weights according to the MB partition modes. Let $f_m(B_{ij})$ denote the partition mode of $B_{ij}$. Then the weights are computed as follows:

$$W_{ij} = \begin{cases} 1 & \text{if } f_m(PB_{ij}) \text{ is } 16\times 16 \\ 2 & \text{if } f_m(PB_{ij}) \text{ is } 16\times 8 \text{ or } 8\times 16 \\ 3 & \text{if } f_m(PB_{ij}) \text{ is } 8\times 8 \\ 4 & \text{if } f_m(PB_{ij}) \text{ is } 8\times 4 \text{ or } 4\times 8 \\ 5 & \text{if } f_m(PB_{ij}) \text{ is } 4\times 4, \end{cases} \tag{3}$$

where $PB_{ij}$ denotes the parent block of the $4\times 4$ pseudo-block $B_{ij}$.

*3) Foreground Segmentation:* This process is to separate the foreground regions from background in the normalized MV field. To this end, we adopt the thresholding strategy to produce the foreground segments, *i.e.* the block $B_i$ inside the ROI is labeled as foreground if its MV is larger than the preset threshold $T_f$. A binary region of interest (ROI) is also applied to mask the foreground segments outside the region of interest. Figure 3 presents an example segmentation result. It can be observed that the moving vehicles can be roughly localized by the segmented foreground regions although a part of backgrounds may also be involved.

*4) Perspective Normalization:* In the frames of surveillance video, further vehicles appear smaller than nearer ones due to the perspective effects. Consequently, the features extracted from the same object with different scene depths would be diverse. To deal with such an issue, the perspective normalization is usually performed. Practically, each block is associated with a weight, and larger weights are assigned to the further vehicle candidates.

The perspective effect is almost fixed for a certain camera or workstation. Thus we only need to periodically sample the video frames and then update the perspective normalization map (denoted by $S$ with each block one value). In this paper, we first adopt the method in [24] to compute the perspective normalization map in the pixel domain, and then transform it into the desired map $S$ in the MV filed by down-sampling.

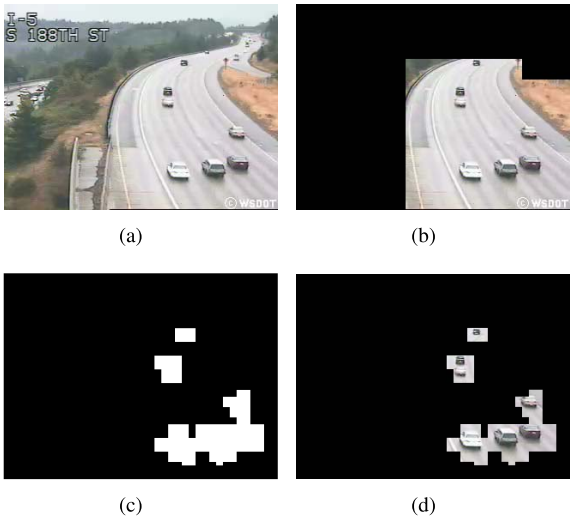



Fig. 3. Foreground segmentation: (a) original frame, (b) ROI, (c) foreground mask, and (d) foreground image.

### C. Feature Extraction

We elaborate on details of feature extraction in this subsection. Ideally, the features should capture the significant information associated with vehicle count or density. To this end, we develop a new batch of low-level features, which cover the size, shape, motion, and texture.

*1) Size:* The size features can capture the magnitude of holistic foreground segment. Here we particularly use two metrics, *i.e.* area and perimeter length.

- *Area*: It is defined as the total number of blocks belonging to the segmented foreground. This feature is calculated from the perspective normalization map $S$ and MB type weights $W_{ij}$, *i.e.*

$$f_a = \sum_{B_{ij}\in\mathcal{F}} W_{ij}\cdot S_{ij}, \tag{4}$$

  where $\mathcal{F}$ represents the foreground area.
- *Perimeter length*: It is defined as the total number of blocks lying on the perimeter of foreground segment. Formally, this feature is weighted using MB type weights $W_{ij}$ and square root of perspective normalization map $S$ as in [24]:

$$f_l = \sum_{B_{ij}\in\mathcal{P}} W_{ij}\cdot\sqrt{S_{ij}}, \tag{5}$$

  where $\mathcal{P}$ denotes the set of perimeter blocks.

*2) Shape:* Aside from the *Perimeter length*, which captures the global properties of the segments, the orientation of perimeter blocks also carries significant shape information due to presenting some local and internal pattern. In this paper, therefore, we define the shape feature as an orientation histogram of perimeter blocks, where eight bins are used.

For the block $B_{ij}$, the orientation $o_{ij}$ and magnitude $m_{ij}$ are calculated as follows:

$$o_{ij} = \tan^{-1}\{g_y(\tilde{V}_{ij})/g_x(\tilde{V}_{ij})\}$$
$$m_{ij} = \sqrt{g_x(\tilde{V}_{ij})^2 + g_y(\tilde{V}_{ij})^2}, \quad (6)$$

where $g_x(\tilde{V}_{ij})$ and $g_y(\tilde{V}_{ij})$ denote the horizontal and vertical components of $\tilde{V}_{ij}$, respectively. In addition, the voting weight of $B_{ij}$ is adjusted by $W$ and $S$ in computing the histogram, which is set as $(m_{ij} \cdot W_{ij} \cdot \sqrt{S_{ij}})$.

*3) HOMV:* MVs reflect the motion orientation and magnitude of objects represented by MBs. In this paper, we propose to use the Histogram of Oriented Motion Vector (HOMV) feature to present such information. The calculation of HOMV is similar to the shape feature, except for the MB range and weight. To be specific, all foreground MBs are used for HOMV, and the voting weight for $B_{ij}$ is set as $(m_{ij} \cdot W_{ij} \cdot S_{ij})$.

*4) Texture:* The texture features are strongly correlated with vehicle density in traffic scenes. Compared with the scenes of low density, the scenes of high density tend to present finer patterns [43]. So we extract the texture feature to capture such a clue. In object counting, two texture features are widely used, *i.e.* Gray-level co-occurrence matrix (GLCM) [24], [44] and local binary pattern (LBP) [45]. In this work, we particularly consider LBP due to its simplicity and effectiveness. We then propose a compressed-domain LBP feature based on MV which is similar to LBP in pixel domain. Actually, LBP has a variety of extensions and modifications developed for better robustness, discriminativeness, and applicability, *e.g.* VLBP [46], CS-LBP [47], SILTP [48], and MRELBP [49]. After investigating these descriptors, we finally use CS-LBP in this paper since it trades-off better efficiency and counting performance. The underlying idea of CS-LBP is to compare the pairs of pixels in the centered symmetric directions instead of comparing the central pixel to its neighbors in the original LBP. This halves the number of comparisons for the same number of neighbors, and finally produces 16 ($2^4$) binary patterns different from 256 ($2^8$) of the original LBP. Formally, the LBP operator of the target MB $B_{ij}$ is defined as:

$$\mathrm{LBP}_{ij} = \sum_{i=0}^{(P/2)-1} s(d(\tilde{V}_i, \tilde{V}_{i+(P/2)})) \cdot 2^i, \quad (7)$$

where $P = 8$ represents the number of neighboring MBs of $B_{ij}$, and $d(\cdot)$ is a distance metric function to measure the similarity between two MVs, which is defined as:

$$d(\tilde{V}_i, \tilde{V}_j) = \exp\left\{-\frac{\|\tilde{V}_i - \tilde{V}_j\|^2}{\|\tilde{V}_i\|^2 + \|\tilde{V}_j\|^2}\right\}. \quad (8)$$

In addition, $s(\cdot)$ is a sign function:

$$s(x) = \begin{cases} 1 & x \geq T_s \\ 0 & x < T_s. \end{cases} \quad (9)$$

$T_s$ is a threshold.

The final texture feature in this paper is the histogram of the LBP outputs accumulated over all foreground MBs. All of proposed features are then concatenated together to form the feature vector of one frame.

Fig. 4. Example frames from the UCSD highway traffic dataset. The sample frames present various vehicle densities: light (top), medium (middle), and heavy (bottom).

Fig. 5. Example frames from the US highway 101 traffic dataset. The sample frames present the transition between uncongested and congested conditions, and full congestion.

### *D. Counting With Spatial Regression*

Now we explain the vehicle counting method that is used to estimate the number of vehicles in a frame. In the previous works, the regression based methods have shown promising performance [10], [50]. Thus we adopt the regression as the base model. Note that this work is the first research attempt to the vehicle counting in compressed domain, while the previous works mainly focus on the crowd counting in pixel domain, *e.g.*, on the UCSD pedestrian dataset [24] or Mall dataset [25]. For crowd counting, the foreground areas vary nearly linear with the number of people. However, such a correlation does not hold for vehicle counting due to larger variation of vehicles, especially in compressed domain.

To intuitively show the characteristics of traffic scenes, Figure 4 and Figure 5 provide some realistic highway images, and Figure 6 demonstrates the relationship between the vehicle count and foreground area. It can be seen that the correlation is much more complicated than simple linearity. In principle, the major factors causing such complication include the broad variation of vehicle appearance, inaccurate information provided by compressed videos, and wide visual field of surveillance cameras. To tackle these challenges, we propose a Hierarchical Classification based Regression (HCR) model in this work. HCR utilizes the local correlation which is valid if the vehicle density[1] only varies within a small range. Namely, the traffic scenes containing different numbers of vehicles would present the density-specific patterns. This assumption is reasonable in practice, as validated by the results in Figure 6.

[1]It is equivalent to the vehicle count for a fixed camera vision.

The HCR model is illustrated in Figure 1. Specifically, we first apply a multi-class classifier to divide the traffic scenes into $k$ categories representing different ranges of vehicle density. According to the patterns shown in Figure 6(a), $k = 3$ is adopted for the UCSD dataset in our experiments (*i.e. heavy*, *medium*, and *light*), which is in accord with the setting in the original work [51]. Indeed, we conduct the comparative experiments with different settings (*e.g.*, $k = 2$ and $k = 4$) and in practice the current setting achieves the best performance. Similarly, $k$ is set to 2 for the US101 dataset according to the patterns shown in Figure 6(b). For the purpose of vehicle counting, the estimation error may be rather significant if the input traffic scene is misclassified. Thus we introduce the second-layer classifiers to deal with the ambiguity cases. The second layer together with the first-layer classifiers form a soft-segmented multiple classifiers. Here the $(k-1)$ binary classifiers are deployed in the second layer, and each of them takes charge of one boundary area. When a new traffic scene arrives, it would be separately classified by both classifier layers, and two or three confidence scores are produced. In particular, only the samples with the scores in the second layer larger than a given threshold $T_c$ are identified to belong to the boundary area. As a result, the input scene is finally assigned to one of the $(2k-1)$ classes.

In our implementation, we adopt the SVM classifier with radial basis function (RBF) kernel [52] to perform the classification. In addition, we use Gaussian Process Regression (GPR) [53], which does not require any prior assumptions, as the regression model to estimate the number of vehicles for each class. It is worth pointing out that by combining different covariance functions, *e.g.*, linear, rational quadratic, and squared-exponential, GPR has the flexibility to encode different assumptions about the function we wish to learn. In this work, the following covariance function [24] is employed:

$$\mathcal{K}(\mathbf{x}_i, \mathbf{x}_j) = a_0 + a_1 \mathbf{x}_i^T \mathbf{x}_j + a_2 \exp\left(-\frac{|\mathbf{x}_i - \mathbf{x}_j|^2}{2a_3}\right) + \delta_{ij} a_4. \tag{10}$$

Here $\delta_{ij}$ is a sign function with 1 if $i = j$ and 0 otherwise, and $\theta = (a_0, a_1, a_2, a_3, a_4)$ is the hyper-parameters, which defines the covariance function. The first two terms capture the linear trend, the third captures local non-linearities, and the last one models the observation noise.

### *E. Refinement With Temporal Regression*

In the proposed counting method, only one single frame is used to estimate the number of vehicles contained by that frame in the video stream and the inherent temporal information in videos is ignored. Consequently, the counting results may fluctuate since the information represented by the metadata of compressed videos is not enough to model the realistic traffic scenes. To further enhance the counting performance, we propose a temporal refinement method to improve the results, which exploits the prior of temporal continuity of vehicle counts in video stream. In fact, each running vehicle in highway will travel within the visual field of a specified camera for a while, and thus the number of vehicles will vary slowly for the sequent frames of video stream.

To model the local temporal consistency of the counting results in video stream, we particularly adopt the robust locally weighted regression method named LOESS [54], [55] in this paper. LOESS is a simple and flexible data processing method, which can build up a function describing the deterministic part of variations in the data. Specifically, for a sequence of consecutive counting results, a low-degree polynomial is used to fit a subset of data around each estimation point, and then the smoothed value for the point is used as the final vehicle count. Such value is generated in practice by evaluating the local polynomial. Here the polynomial is fitted using the weighted least squares: giving larger weights to points near the target point (whose value is being estimated) and smaller weight to points further away. In LOESS, a smoothing parameter, which determines how much data to use for fitting each local polynomial, and the degree of the local polynomial need to be specified. In our implementation, we adopt the LOESS provided by the Curve Fitting Toolbox[2] and the quadratic polynomial is adopted for all samples. Formally, Algorithm 1 describes the the local regression process for single data sample, and Algorithm 2 provides a robust version which includes an additional calculation of robust weights for alleviating the distortion of outliers.

**Algorithm 1:** The Locally Weighted Regression

**Input:** The estimation value $c$ and its neighbors $c_i$;
**Output:** The smoothed value $\tilde{c}$;

1: Compute the distance $d(c)$ from $c$ to the furthest neighbor.
2: Compute the regression weights for each data point $c_i$.

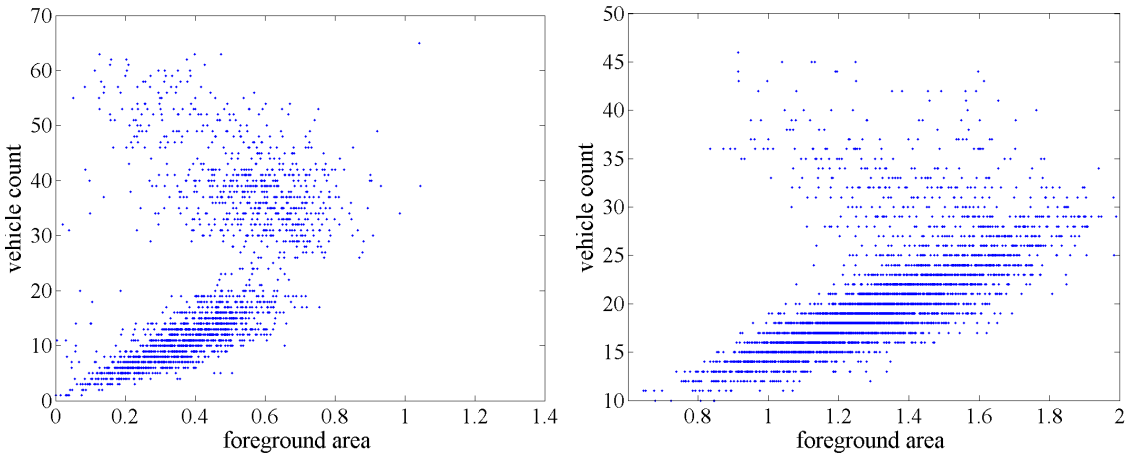
$$w_i = \left(1 - \left|\frac{c - c_i}{d\,(c)}\right|^3\right)^3,$$

3: Perform weighted linear least-squares regression with a second degree polynomial.
4: Compute the smoothed value $\tilde{c}$ through the weighted regression model.
5: **return** $\tilde{c}$;

## IV. Experiment

There are three types of temporally interleaved frames in H.264 bitstream [6]: I-frame, P-frame, and B-frame. The I-frame is absolutely intra-coded, the P-frame is motion compensated in the forward direction from I-frame or other P-frame, and the B-frame is motion compensated in both forward and backward directions. In our experiments, P-frames and I-frames are used since the consecutive P-frames can provide continuous motion information. All videos are encoded using the H.264/AVC JM v.18.6 encoder.[3] We use the same frame features extracted in Section III-C for both classification and counting. As for the thresholds, we

[2] http://www.mathworks.com/products/curvefitting/
[3] http://iphome.hhi.de/suehring/tml/




(a) (b)

Fig. 6. Correspondence between vehicle count and foreground size: (a) UCSD, (b) US101. The correlation is quite complicated compared to the simple linearity, while the local correlation is approximately held if the vehicle density varies within a small range.

**Algorithm 2:** The Robust Locally Weighted Regression

**Input**: The original estimation value $c$ and its neighbors $c_i$; The smoothed value $\tilde{c}$ and its neighbors $\tilde{c}_i$ from Algorithm 1; The number of samples $N$;

**Output**: The smoothed value $\tilde{c}$;

$n = 5$;
**for** $k = 1; k \leq n; k++$ **do**
  *// Compute the residual of each data point*
  **for** $i = 1; i \leq N; i++$ **do**
    $r_i = c_i - \tilde{c}_i$;
  *// Compute the median absolute deviation of the residuals*
  $m_{ad} = \text{median}(|r|)$;
  *// Compute the robust weights for each data point*
  **for** $i = 1; i \leq N; i++$ **do**
    **if** $|r_i| < 6m_{ad}$ **then**
      $w_i = (1 - (r_i/6m_{ad})^2)^2$;
    **else**
      $w_i = 0$;
  Perform weighted linear least-squares regression with a second degree polynomial.
  Update the smoothed value $\tilde{c}$ through the weighted regression model.
return $\tilde{c}$;

set $T_f$ as the minimum value of vehicle motion (*i.e.*, 1) to capture all the vehicle motions in traffic scenes since MVs are the key component of the proposed features. $T_s$ and $T_c$, which are used to measure the MV similarity and classification confidence respectively, are set as 0.9 and 0.7 in the experiments. These values are empirically determined by investigating the counting performance for different settings from 0.5 to 1, and actually the variation of settings only has slight influence to the final performance.

### A. Dataset

As there is no benchmark database for highway vehicle counting, we collected some published highway video sequences and constructed the counting dataset.

*1) UCSD:* The UCSD highway traffic dataset [51] was originally built for the traffic scene classification purpose. This dataset consists of 254 video sequences of daytime highway traffic in Seattle and Washington. Each video contains 42 to 52 frames recorded at 10 frames per second (fps) and the resolution is $320 \times 240$ pixels. Figure 4 provides some example frames. Such a dataset is challenging due to containing diverse traffic patterns, *e.g.*, covering the light, medium, and heavy congestion with various weather conditions (clear, overcast, and raining).

To construct the counting dataset, we first select a region of interest (ROI) in the traffic scene (see Figure 3(b)). Then we extract 8 samples from each video every 5 frames, *i.e.*, the 5th, 10th, 15th, 20th, 25th, 30th, 35th, and 40th frames. Finally, we manually label these frames by marking the central points of vehicle bodies. As a result, a total of 42, 859 vehicles in the 2032 frames are labeled, which cover all representative traffic situations in the UCSD dataset.

As for the classification, we define the dense categories by counting the labeled vehicles. Specifically, the samples containing less than 20 vehicles are categorized as *light*, the ones between 20 and 40 are as *medium*, and the rests are as *heavy*. Besides, we define the boundary area of *light* and *medium* as the range of [16, 24], and that of *medium* and *heavy* is [36, 44].

*2) US101:* We also collected 45 minutes videos obtained from the US Highway 101 dataset [56]. The dataset covers an area in Los Angeles, CA, approximately 640m in length with five mainline lanes and a sixth auxiliary lane providing highway entrance and exit. The full videos in the dataset are recorded by eight synchronized digital video cameras at 10 frames per second (fps) with the resolution of $640 \times 480$. In this paper, we constructed the counting dataset using the 45 minute videos recorded by the camera 4. The videos contain the transition periods between uncongested and congested conditions, and the peak period with full congestion. Figure 5 provides some example frames.

In building the counting dataset, we first divide the videos into the sequences containing nearly 50 frames, and randomly select 300 sequences to balance the traffic patterns in the

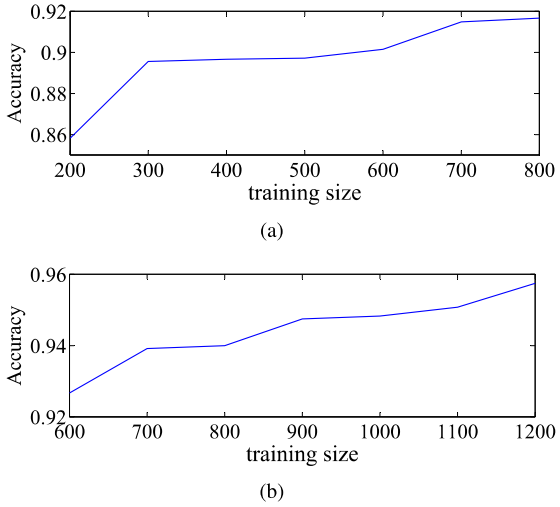



Fig. 7. Classification accuracies for different training sizes: (a) UCSD, (b) US101.

dataset. Then the region consists of five mainline lanes and an auxiliary lane is selected as ROI, and 2400 frames from the 300 video sequences are labeled manually. According to Figure 6(b), the traffic density of US101 highway dataset is classified to two categories, where the samples containing less than 28 vehicles are categorized as *uncongested* and the rests belong to *congested*. For HCR, the boundary range of *uncongested* and *congested* is from 24 to 32.

### B. Classification Results

To evaluate the performance of the traffic density classification, we conduct experiments on the labeled UCSD and US101 highway counting datasets. Here the feature of one frame for classification is generated by averaging the features of five consecutive neighboring frames in order to improve the robustness. For UCSD highway dataset, the samples are randomly split into the training set containing 800 samples and the test set holding the remaining 1232 samples, such that different traffic patterns and weather conditions are balanced in the training and test sets. We vary the number of training samples from 200 to 800 with an interval of 100 to investigate its influence to the performance. For each training setting, we repeat the experiment for five times, and report the mean classification accuracy. Figure 7(a) gives the results. It can be seen that the classification accuracy is around 90% and increases as more training samples are used. Similarly, the 2400 samples of US101 highway dataset are randomly divided into half-half for training and testing respectively, and Figure 7(b) shows the results. Evidently, the proposed features are also accurate and robust for representing the large-varying traffic scenes.

The UCSD dataset was built for classification, and each video sequence is labeled as *light*, *medium*, or *heavy*. We also compare the proposed method with the pixel-domain methods [51], [57]. For fair comparison, the same experimental settings are used, and the average feature on the whole video clips is adopted as the video representation. Finally, our proposed method achieves the mean classification accuracy of 94.22%, which is very close to the best performance of 94.50% achieved by [51] and 95.28% by [57]. These results clearly show that the features extracted from encoded videos are discriminative and robust enough for classification, *i.e.* the compressed-domain method is rather competitive to the pixel-domain ones.

### C. Counting Results

For vehicle counting, we adopt the mean-absolute-error (MAE) to measure the performance, which represents the difference between the predicted counts and the ground truth. Here we conduct multiple experiments with different combinations of features in order to demonstrate the effect of each feature. We also compare the proposed method against the pixel-domain regression methods in [24] and [50]. Specifically, in our implementation, the mixture of dynamic textures used in [24] is adopted as the motion segmentation method for both [24] and [50]. For [50], the holistic features include size, shape, edges, and keypoints, and the GPR model is used for evaluation. As in the classification experiment, 800 and 1200 samples are selected as the training set for the UCSD and US101 datasets, respectively, and each experiment is repeated for five times. It is worth noting that, the split strategy is done by selecting video sequences rather than samples, and the samples in the selected video sequences form the training set. Through this way, we can remove the impact that the training and test samples may from the same video sequence, although the time gap between samples in the same sequence is big enough.

Moreover, we compare the proposed method with the state-of-art vehicle counting model using deep learning in [32]. We choose it as the baseline as the codes[4] are released for fair comparison. For both two datasets, following the experimental setting in [32], we randomly extract 800 image patches of $115 \times 115$ pixels together with the ground truth, then resize each sample to $72 \times 72$ pixels, and finally augment the training data by flipping. Here the Gaussian parameter for generating the groundtruth density maps is set as $\sigma = 15$. Besides, the perspective map of UCSD dataet is used. During training, the batch normalization [58] is applied to all convolutional and fully connected layers (except for the output layers).

Table I and Table II report the resulting MAEs on the UCSD and US101 datasets. It can be seen that the counting performance is consistently improved as more features are employed. This demonstrates that each proposed feature is contributive to vehicle counting. In particular, the more efficient CS-LBP feature proposed in this paper achieves comparable counting performance to our previous work [8]. In addition, the performance achieved by the proposed method is similar to those of the conventional pixel-domain methods [24], [50]. For the deep learning based vehicle counting method [32], we report the results of the Counting CNN and best performing Hydra CNN 3s models. The results show that though Hydra CNN with 3 scales outperforms all other methods not using deep learning, the proposed method achieves comparative accuracy

[4]https://github.com/gramuah/ccnn

TABLE I

COMPARISON OF COUNTING PERFORMANCE FOR DIFFERENT METHODS AND FEATURE SETS ON THE UCSD DATASET, WHERE THE *Red* AND *Blue* DENOTE THE FIRST AND SECOND BEST PERFORMANCE

| Number of training samples | 200 | 300 | 400 | 500 | 600 | 700 | 800 |
|---|---|---|---|---|---|---|---|
| size | 3.981 | 3.747 | 3.719 | 3.665 | 3.577 | 3.352 | 3.332 |
| size + shape | 3.865 | 3.639 | 3.576 | 3.474 | 3.355 | 3.179 | 3.173 |
| size + shape + HOMV | 3.820 | 3.618 | 3.556 | 3.448 | 3.310 | 3.110 | 3.111 |
| HCR (with all features) [8] | 3.639 | 3.483 | 3.420 | 3.332 | 3.142 | 2.956 | 2.938 |
| HCR (with all features) | 3.583 | 3.537 | 3.398 | 3.323 | 3.176 | 2.961 | 2.928 |
| HCR-LOESS | **2.963** | **2.923** | **2.868** | **2.775** | **2.687** | **2.525** | **2.446** |
| Chan *et al.* [24] | 4.140 | 3.581 | 3.439 | 3.415 | 3.229 | 3.020 | 2.970 |
| Ryan *et al.* [50] | 3.866 | 3.443 | 3.432 | 3.268 | 3.170 | 2.923 | 2.917 |
| Counting CNN [32] | 2.986 | 2.997 | 2.893 | 2.866 | 2.729 | 2.682 | 2.634 |
| Hydra 3s [32] | **2.468** | **2.365** | **2.214** | **2.261** | **2.209** | **2.042** | **2.029** |

TABLE II

COMPARISON OF COUNTING PERFORMANCE FOR DIFFERENT METHODS AND FEATURE SETS ON THE US101 DATASET, WHERE THE *Red* AND *Blue* DENOTE THE FIRST AND SECOND BEST PERFORMANCE

| Number of training samples | 600 | 700 | 800 | 900 | 1000 | 1100 | 1200 |
|---|---|---|---|---|---|---|---|
| size | 1.876 | 1.526 | 1.466 | 1.414 | 1.411 | 1.364 | 1.353 |
| size + shape | 1.751 | 1.486 | 1.415 | 1.354 | 1.342 | 1.303 | 1.295 |
| size + shape + HOMV | 1.704 | 1.444 | 1.363 | 1.301 | 1.286 | 1.262 | 1.256 |
| HCR (with all features) [8] | 1.643 | 1.396 | 1.318 | 1.259 | 1.236 | 1.207 | 1.188 |
| HCR (with all features) | 1.641 | 1.398 | 1.319 | 1.259 | 1.227 | 1.205 | 1.194 |
| HCR-LOESS | 1.535 | 1.289 | 1.206 | 1.150 | 1.131 | **1.112** | 1.097 |
| Chan *et al.* [24] | 1.589 | 1.331 | 1.212 | 1.153 | **1.124** | 1.115 | 1.086 |
| Ryan *et al.* [50] | 1.551 | 1.317 | 1.209 | **1.146** | **1.124** | 1.113 | **1.058** |
| Counting CNN [32] | **1.199** | **1.180** | **1.178** | 1.169 | 1.166 | 1.159 | 1.121 |
| Hydra 3s [32] | **0.995** | **0.989** | **0.972** | **0.977** | **0.941** | **0.954** | **0.924** |

with the Counting CNN model. Thus we consider the proposed method is very competitive.

We specially evaluate the proposed refinement method by applying the temporal regression (LOESS) to samples in the same video sequence. Here the smoothing parameter is set to 0.5, *i.e.* the value at each point is refined by operating twenty frames around. The quantitative results for the proposed HCR model are shown in Table I and Table II. Evidently, the counting performance is improved considerably, especially on the UCSD highway dataset. Moreover, we compare the effect of the proposed temporal refinement for the compressed-domain method and conventional pixel-domain methods, where the UCSD dataset is used. Figure 8 provides the results. It is shown that LOESS can always boost the counting performance. Particularly, LOESS is more effective for the compressed-domain method since the information in compressed domain is much noisier due to the inaccuracy of encoding metadata. To provide an intuitive analysis, Figure 9 plots the estimation results before and after applying LOESS against the ground truth, where a number of sequences on the UCSD and US101 datasets are used. These figures provide the qualitative evidence that the proposed temporal refinement actually removes outliers to boost the counting performance.

We further evaluate the proposed HCR by comparing it with the single Gaussian model (GP), one-layer multi-regression model (MGR), and the ideal version of HCR that adopts the ground-truth classes instead of the predicted ones. Figure 10 gives intuitive performance comparison for different models. Evidently, multiple regressions generally outperform the single regression, and introducing second-layer classifiers in HCR puts the counting performance towards the optimal.

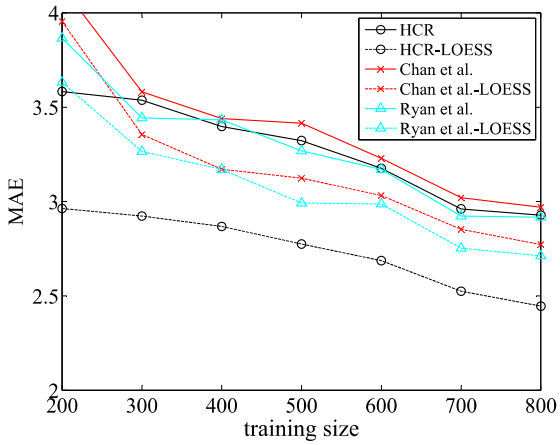



Fig. 8. Effects of the proposed temporal refinement for different counting methods on the UCSD dataset.

### *D. Run-Time Analysis*

As described above, the proposed method consists of four main components: preprocessing, feature extraction,

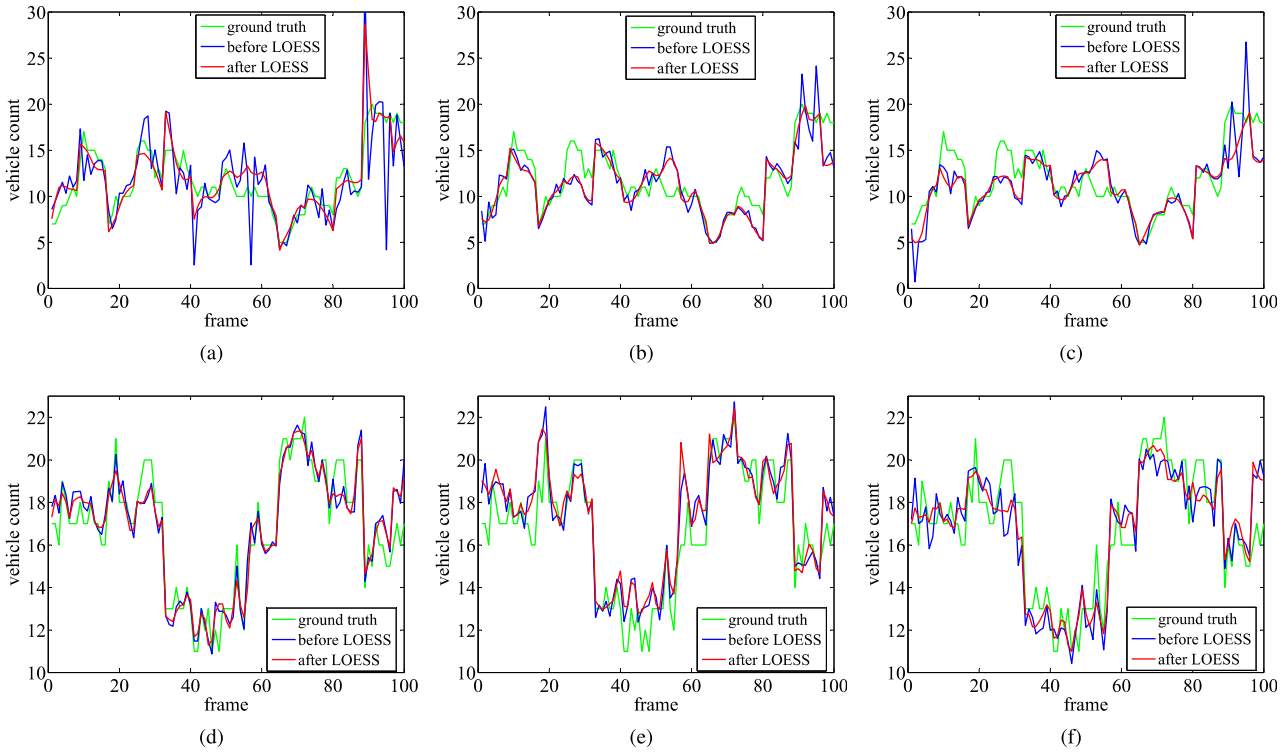



Fig. 9. The estimated vehicle counts before and after LOESS against the ground truth on UCSD (top) and US101 (bottom). Best viewed in the color version. (a) HCR. (b) Chan *et al.* [24]. (c) Ryan *et al.* [50]. (d) HCR. (e) Chan *et al.* [24]. (f) Ryan *et al.* [50].

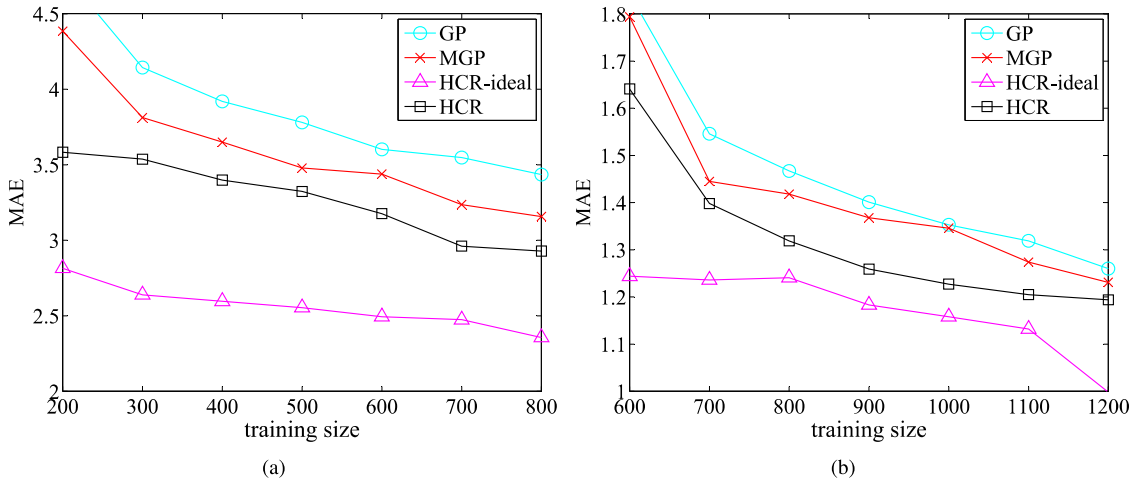



Fig. 10. Comparison of MAE for different models: (a) UCSD, (b) US101. Here GP denotes the single GPR model, MGP represents the one-layer multi-regression model, and HCR-ideal is the ideal version of HCR that adopts the ground truth as the classification results.

classification with SVM, and GPR. For both [24] and [50], in addition to the full decoding and reconstruction of video frames, the counting frameworks are mainly composed of feature extraction and GPR. Particularly, the run-time complexity of SVM using an RBF kernel in our method is $O(N_{SV} \times d)$, where $N_{SV}$ is the number of support vectors and $d$ is the input dimensionality. For a training set containing $N$ cases, the complexity of making a prediction with gaussian process is $O(N^2)$. Thus the complexity of HCR is similar to the pixel-domain methods since they need the same Gaussian processes. However, the proposed method is faster in practice due to more efficient preprocessing and feature extraction. First, it avoids fully decoding videos and frame reconstruction. Second, it processes less data compared to the pixel-domain methods as the MVs is significantly less than pixels.

Overall, the resolution of the normalized MV field in compressed domain is one-sixteenth of the reconstructed pixel image. In addition to the computational saving of video decoding, we experimentally compare the run-time of preprocessing and feature extraction in the proposed method against the pixel-domain methods. Here the US101 dataset is selected and the images have a resolution of 640 × 480. The same configurations are used for different methods, *i.e.*, all the codes are implemented in Matlab and a single CPU

core (Intel Xeon CPU E5-2620 @ 2.00GHz) is employed. As expected, the average run-time of our proposed method is much less than that of the pixel-domain methods. Specifically, our method achieves about 110 fps on US101, while the methods in [24] and [50] result in about 9 fps and 6 fps respectively.

## V. Conclusion

In this work, we proposed a novel highway vehicle counting method in compressed domain with aims of achieving comparable estimation performance with the pixel-domain methods. Specifically, we first developed new low-level features by utilizing the encoding metadata of compressed videos. Then we proposed a hierarchical classification based regression model (HCR) to estimate the number of vehicles in a frame, and a temporal regression method to refine the counting results. Finally, we verified the effectiveness of the proposed method through the experimental evaluation. This work shows that the compressed-domain method is promising for the real-world video surveillance systems and demonstrates the advantages of low computational cost, convenient deployment, and competitive performance. Although the proposed method works well even for congested traffic scenes, the counting performance may be deteriorated as the number of stationary vehicles increases, since the MVs in video bitstreams would be vanished significantly. In addition, the high-varying sizes of different types of vehicles could affect the counting accuracy. In the future, we plan to detect the situations of traffic scenes and explore the influence of vehicle types for achieving more robust counting results.

## References


[1] E. Bas, A. M. Tekalp, and F. S. Salman, "Automatic vehicle counting from video for traffic flow analysis," in *Proc. IEEE IVS*, Jun. 2007, pp. 392–397.

[2] M. Liang, X. Huang, C.-H. Chen, X. Chen, and A. Tokuta, "Counting and classification of highway vehicles by regression analysis," *IEEE Trans. Intell. Transp. Syst.*, vol. 16, no. 5, pp. 2878–2888, Oct. 2015.

[3] S.-C. Huang and B.-H. Chen, "Automatic moving object extraction through a real-world variable-bandwidth network for traffic monitoring systems," *IEEE Trans. Ind. Electron.*, vol. 61, no. 4, pp. 2099–2112, Apr. 2014.

[4] S. H. Khatoonabadi and I. V. Bajic, "Video object tracking in the compressed domain using spatio-temporal Markov random fields," *IEEE Trans. Image Process.*, vol. 22, no. 1, pp. 300–313, Jan. 2013.

[5] R. V. Babu, M. Tom, and P. Wadekar, "A survey on compressed domain video analysis techniques," *Multimedia Tools Appl.*, vol. 75, no. 2, pp. 1043–1078, Jan. 2016.

[6] T. Wiegand, G. J. Sullivan, G. Bjontegaard, and A. Luthra, "Overview of the H.264/AVC video coding standard," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 13, no. 7, pp. 560–576, Jul. 2003.

[7] B. Tian *et al.*, "Hierarchical and networked vehicle surveillance in ITS: A survey," *IEEE Trans. Intell. Transp. Syst.*, vol. 16, no. 2, pp. 557–580, Apr. 2015.

[8] X. Liu, Z. Wang, J. Feng, and H. Xi, "Highway vehicle counting in compressed domain," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, Jun. 2016, pp. 3016–3024.

[9] V. Lempitsky and A. Zisserman, "Learning to count objects in images," in *Proc. Adv. Neural Inf. Process. Syst.*, 2010, pp. 1324–1332.

[10] C. C. Loy, K. Chen, S. Gong, and T. Xiang, "Crowd counting and profiling: Methodology and evaluation," in *Proc. Modeling, Simulation Vis. Anal. Crowds*, vol. 11. Oct. 2013, pp. 347–382.

[11] Z. Sun, G. Bebis, and R. Miller, "On-road vehicle detection: A review," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 28, no. 5, pp. 694–711, May 2006.

[12] P. Dollár, C. Wojek, B. Schiele, and P. Perona, "Pedestrian detection: An evaluation of the state of the art," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 34, no. 4, pp. 743–761, Apr. 2012.

[13] P. F. Felzenszwalb, R. B. Girshick, D. McAllester, and D. Ramanan, "Object detection with discriminatively trained part-based models," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 32, no. 9, pp. 1627–1645, Sep. 2010.

[14] N. Dalal and B. Triggs, "Histograms of oriented gradients for human detection," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, Jun. 2005, pp. 886–893.

[15] H. T. Niknejad, A. Takeuchi, S. Mita, and D. McAllester, "On-road multivehicle tracking using deformable object model and particle filter with improved likelihood estimation," *IEEE Trans. Intell. Transp. Syst.*, vol. 13, no. 2, pp. 748–758, Jun. 2012.

[16] A. Geiger, M. Lauer, C. Wojek, C. Stiller, and R. Urtasun, "3D traffic scene understanding from movable platforms," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 36, no. 5, pp. 1012–1025, May 2014.

[17] B. Pepikj, M. Stark, P. Gehler, and B. Schiele, "Occlusion patterns for object class detection," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, Jun. 2013, pp. 3286–3293.

[18] S. Ren, K. He, R. Girshick, and J. Sun, "Faster R-CNN: Towards real-time object detection with region proposal networks," in *Proc. Adv. Neural Inf. Process. Syst.*, 2015, pp. 91–99.

[19] W. Liu *et al.*, "SSD: Single shot MultiBox detector," in *Proc. Eur. Conf. Comput. Vis.*, 2016, pp. 21–37.

[20] J. Redmon and A. Farhadi, "YOLO9000: Better, faster, stronger," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, Jul. 2017, pp. 7263–7271.

[21] Y. Zhou, L. Liu, L. Shao, and M. Mellor, "DAVE: A unified framework for fast vehicle detection and annotation," in *Proc. Eur. Conf. Comput. Vis.*, 2016, pp. 278–293.

[22] V. Rabaud and S. Belongie, "Counting crowded moving objects," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, Jun. 2006, pp. 705–711.

[23] R. Zhao and X. Wang, "Counting vehicles from semantic regions," *IEEE Trans. Intell. Transp. Syst.*, vol. 14, no. 2, pp. 1016–1022, Jun. 2013.

[24] A. B. Chan, Z.-S. J. Liang, and N. Vasconcelos, "Privacy preserving crowd monitoring: Counting people without people models or tracking," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, Jun. 2008, pp. 1–7.

[25] K. Chen, C. C. Loy, S. Gong, and T. Xiang, "Feature mining for localised crowd counting," in *Proc. Brit. Mach. Vis. Conf.*, 2012, pp. 21-1–21-11.

[26] C. Zhang, H. Li, X. Wang, and X. Yang, "Cross-scene crowd counting via deep convolutional neural networks," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, Jun. 2015, pp. 833–841.

[27] A. C. Davies, J. H. Yin, and S. A. Velastin, "Crowd monitoring using image processing," *Electron. Commun. Eng. J.*, vol. 7, no. 1, pp. 37–47, Feb. 1995.

[28] B. Tan, J. Zhang, and L. Wang, "Semi-supervised elastic net for pedestrian counting," *Pattern Recognit.*, vol. 44, nos. 10–11, pp. 2297–2304, Oct./Nov. 2011.

[29] W. Xia, J. Zhang, and U. Kruger, "Semisupervised pedestrian counting with temporal and spatial consistencies," *IEEE Trans. Intell. Transp. Syst.*, vol. 16, no. 4, pp. 1705–1715, Aug. 2015.

[30] C. C. Loy, S. Gong, and T. Xiang, "From semi-supervised to transfer counting of crowds," in *Proc. IEEE Int. Conf. Comput. Vis.*, Dec. 2013, pp. 2256–2263.

[31] Y. Zhang, D. Zhou, S. Chen, S. Gao, and Y. Ma, "Single-image crowd counting via multi-column convolutional neural network," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, Jun. 2016, pp. 589–597.

[32] D. Oñoro-Rubio and R. J. López-Sastre, "Towards perspective-free object counting with deep learning," in *Proc. Eur. Conf. Comput. Vis.*, 2016, pp. 615–629.

[33] S. Zhang, G. Wu, J. P. Costeira, and J. M. F. Moura, "Understanding traffic density from large-scale Web camera data," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, Jul. 2017, pp. 5898–5907.

[34] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, Jun. 2016, pp. 770–778.

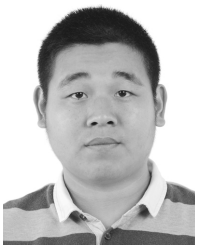
[35] F. Porikli, F. Bashir, and H. Sun, "Compressed domain video object segmentation," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 20, no. 1, pp. 2–14, Jan. 2010.

[36] B. Dey and M. K. Kundu, "Robust background subtraction for network surveillance in H.264 streaming video," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 23, no. 10, pp. 1695–1703, Oct. 2013.

[37] 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.
[38] H. Wang and S.-F. Chang, "A highly efficient system for automatic face region detection in MPEG video," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 7, no. 4, pp. 615–628, Aug. 1997.
[39] S. S. S. Kruthiventi and R. V. Babu, "Crowd flow segmentation in compressed domain using CRF," in *Proc. Int. Conf. Image Process.*, Sep. 2015, pp. 3417–3421.
[40] F. Porikli and X. Li, "Traffic congestion estimation using HMM models without vehicle tracking," in *Proc. IEEE IVS*, Jun. 2004, pp. 188–193.
[41] R. Tusch *et al.*, "Efficient level of service classification for traffic monitoring in the compressed video domain," in *Proc. IEEE Int. Conf. Multimedia Expo*, Jul. 2012, pp. 967–972.
[42] X. Yu, P. Xue, L. Duan, and Q. Tian, "An algorithm to estimate mean vehicle speed from MPEG Skycam video," *Multimedia Tools Appl.*, vol. 34, no. 1, pp. 85–105, Jul. 2007.
[43] A. N. Marana, S. A. Velastin, L. F. Costa, and R. A. Lotufo, "Estimation of crowd density using image processing," in *Proc. IEE Colloq. Image Process. Secur. Appl.*, Mar. 1997, pp. 11-1–11-8.
[44] R. M. Haralick, K. Shanmugam, and I. Dinstein, "Textural features for image classification," *IEEE Trans. Syst., Man, Cybern.*, vol. SMC-3, no. 6, pp. 610–621, Nov. 1973.
[45] W. Ma, L. Huang, and C. Liu, "Advanced local binary pattern descriptors for crowd estimation," in *Proc. IEEE Pacific–Asia Workshop Comput. Intell. Ind. Appl.*, Dec. 2008, pp. 958–962.
[46] G. Zhao and M. Pietikäinen, "Dynamic texture recognition using local binary patterns with an application to facial expressions," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 29, no. 6, pp. 915–928, Jun. 2007.
[47] M. Heikkilä, M. Pietikäinen, and C. Schmid, "Description of interest regions with local binary patterns," *Pattern Recognit.*, vol. 42, no. 3, pp. 425–436, Mar. 2009.
[48] S. Liao, G. Zhao, V. Kellokumpu, M. Pietikäinen, and S. Z. Li, "Modeling pixel process with scale invariant local patterns for background subtraction in complex scenes," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, Jun. 2010, pp. 1301–1306.
[49] L. Liu, S. Lao, P. W. Fieguth, Y. Guo, X. Wang, and M. Pietikäinen, "Median robust extended local binary pattern for texture classification," *IEEE Trans. Image Process.*, vol. 25, no. 3, pp. 1368–1381, Mar. 2016.
[50] D. Ryan, S. Denman, S. Sridharan, and C. Fookes, "An evaluation of crowd counting methods, features and regression models," *Comput. Vis. Image Understand.*, vol. 130, pp. 1–17, Jan. 2015.
[51] A. B. Chan and N. Vasconcelos, "Classification and retrieval of traffic video using auto-regressive stochastic processes," in *Proc. IEEE IVS*, Jun. 2005, pp. 771–776.
[52] C.-C. Chang and C.-J. Lin, "LIBSVM: A library for support vector machines," *ACM Trans. Intell. Syst. Technol.*, vol. 2, no. 3, pp. 27:1–27:27, 2011.
[53] C. E. Rasmussen, *Gaussian Processes for Machine Learning*. Cambridge, MA, USA: MIT Press, 2006.
[54] W. S. Cleveland, "Robust locally weighted regression and smoothing scatterplots," *J. Amer. Statist. Assoc.*, vol. 74, no. 368, pp. 829–836, Dec. 1979.
[55] W. S. Cleveland and S. J. Devlin, "Locally weighted regression: An approach to regression analysis by local fitting," *J. Amer. Statist. Assoc.*, vol. 83, no. 403, pp. 596–610, Sep. 1988.
[56] J. Colyar and J. Halkias, "Us highway 101 dataset," Federal Highway Admin., Oklahoma City, OK, USA, Tech. Rep. FHWA-HRT-07-030, 2007.
[57] K. G. Derpanis and R. P. Wildes, "Classification of traffic video based on a spatiotemporal orientation analysis," in *Proc. IEEE Workshop Appl. Comput. Vis.*, Jan. 2011, pp. 606–613.
[58] S. Ioffe and C. Szegedy, "Batch normalization: Accelerating deep network training by reducing internal covariate shift," in *Proc. Int. Conf. Mach. Learn.*, 2015, pp. 448–456.

**Zilei Wang** (M'13) received the B.S. and Ph.D. degrees in control science and engineering from the University of Science and Technology of China (USTC), Hefei, China, in 2002 and 2007, respectively.

He is currently an Associate Professor with the Department of Automation, USTC, and the Founding Leader of the Vision and Multimedia Research Group. Before joining the National University of Singapore (NUS) as a Faculty Member, he was a Post-Doctoral Research Fellow with NUS. His current research interests include computer vision, pattern recognition, and deep learning.



**Xu Liu** received the B.S. degree in control science and engineering from the University of Science and Technology of China, Hefei, China, in 2013, where he is currently pursuing the Ph.D. degree in control science and engineering. His current research interests are object counting, object detection, and object tracking.

**Jiashi Feng** received the Ph.D. degree from the National University of Singapore (NUS) in 2014. Before joining NUS as a Faculty Member, he was a Post-Doctoral Research Fellow with the University of California at Berkeley, Berkeley, CA, USA. He is currently an Assistant Professor with the Department of Electrical and Computer Engineering, NUS. His research areas include computer vision and machine learning. In particular, he is interested in object recognition, detection, segmentation, robust learning, and deep learning.

**Jian Yang** (M'09–SM'16) received the B.S. and Ph.D. degrees from the University of Science and Technology of China (USTC), Hefei, China, in 2001 and 2006, respectively. From 2006 to 2008, he was a Post-Doctoral Scholar with the Department of Electronic Engineering and Information Science, USTC.

He is currently a Professor with the School of Information Science and Technology, USTC. His research interests include future network, distributed system design, modeling and optimization, multimedia over wired and wireless and stochastic optimization. He received the Lu Jia-Xi Young Talent Award from the Chinese Academy of Sciences in 2009.

**Hongsheng Xi** received the B.S. and M.S. degrees in applied mathematics from the University of Science and Technology of China (USTC), Hefei, China, in 1980 and 1985, respectively.

He is currently a Professor with the Department of Automation, USTC. His current research interests include stochastic control systems, network performance analysis and optimization, wireless communications, and signal processing.