Highway Vehicle Counting in Compressed Domain

Xu Liu[1], Zilei Wang[2], Jiashi Feng[3], Hongsheng Xi[2]
[1,2] Department of Automation, University of Science and Technology of China, Hefei, 230027, China
[3] Department of ECE, National University of Singapore, Singapore
[1]liuxu91@mail.ustc.edu.cn, [2]{zlwang, xihs}@ustc.edu.cn, [3]elefjia@nus.edu.sg

Abstract

This paper presents a highway vehicle counting method in compressed domain, aiming at achieving acceptable estimation performance approaching the pixel-domain methods. Such a task essentially is challenging because the available information (e.g. motion vector) to describe vehicles in videos is quite limited and inaccurate, and the vehicle count in realistic traffic scenes always varies greatly. To tackle this issue, we first develop a batch of low-level features, which can be extracted from the encoding metadata of videos, to mitigate the informational insufficiency of compressed videos. Then we propose a Hierarchical Classification based Regression (HCR) model to estimate the vehicle count from features. HCR hierarchically divides the traffic scenes into different cases according to vehicle density, such that the broad-variation characteristics of traffic scenes can be better approximated. Finally, we evaluated the proposed method on the real highway surveillance videos. The results show that our method is very competitive to the pixel-domain methods, which can reach similar performance along with its lower complexity.

1. Introduction

Estimating the number of on-road vehicles is one of the important tasks in intelligent transportation system (ITS), which can be used to monitor the traffic status and then guide the traffic control and optimization, *e.g.* programing better driving routes through bypassing the congested roads [2]. In this paper, we particularly consider the vehicle counting issue by the means of video analysis. Compared with employing specialized sensors (*e.g.* infrared or inductive loop detector), the visual counting system is more easy to deploy by reusing the roadside cameras, and thus pays lower cost [16].

In a typical traffic surveillance system, the central subsystem connects all terminal cameras through a private network and is expected to constantly receive the surveillance videos. In practice, however, only part of video streams can be simultaneously accessed due to the limitation of network bandwidth [12]. Such partial acquisition is rather tolerable for the purpose of monitoring with the help of free stream switch. But it is inapplicable to the video analysis that needs to process almost all of video streams for fully capturing the traffic status of a wide range. Thus it is naturally considered to conduct the analysis function in the terminal devices (*e.g.* surveillance workstations). These devices are usually equipped with lower configurations than the central servers considering the overall costs. Therefore, the video analysis algorithms are expected to have low computational complexity for fulfilling the real-time requirements.

The video analysis can be conducted in the pixel or compressed domain. Particularly, the pixel-domain methods are to first decode the surveillance videos into huge volume of frame pixels and then operate on the pixels to complete some specific target. Due to involving such decoding procedure and massiveness of processed pixels, the high computational complexity is usually possessed [13]. On the contrary, the compressed-domain methods are to directly operate on the video data of compressed format, which is exactly the original form of storing and transmitting videos [1].

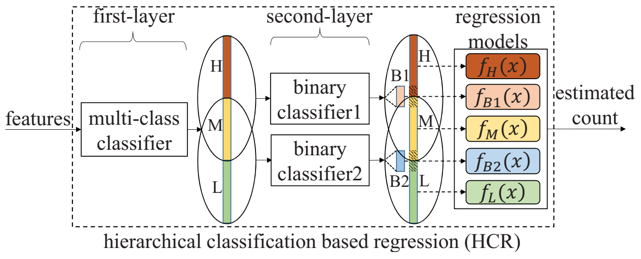

Figure 1. The Hierarchical Classification based Regression (HCR) model. The traffic scenes are firstly classified into three categories: H (heavy), M (medium), and L (light). To tackle the boundary cases produced by the first-layer classifier, *i.e.* samples around the boundary area of two categories are more likely to be misclassified, the second layer classifiers are introduced to distinguish the samples whether located in the boundary area (*i.e.* B1 or B2). Afterwards, regression models are applied to each category.

Hence it is considered that the compressed-domain methods may be more appropriate for large-scale video analysis system. In this paper, we particularly address the vehicle counting issue in compressed domain.

The video analysis methods in compressed domain mainly rely 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 [1]. There are two non-trivial challenges for vehicle counting. First, the critical metadata in compressed videos (*i.e.* motion estimation and compensation vectors) are originally determined from the view of compression efficiency rather than video analysis [30]. Consequently, the features extracted from video bitstreams are probably inaccurate and noisy in describing moving vehicles, besides less available information can be provided compared to the frame pixels. Second, the traffic scenes are usually fast-changing with containing various numbers of vehicles in a broad range, and vulnerable to the external factors (*e.g.* weather conditions, and illumination changes) [28]. These challenges make it quite difficult to accurately model the realistic traffic scenes for vehicle counting.

In this paper, we propose a multi-regression method for highway vehicle counting in compressed domain, with aims of achieving acceptable prediction performance approaching the pixel-domain methods. To our best knowledge, this is the first attempt on this issue. Specifically, we first develop a batch of low-level features to capture the crucial information associated with vehicle count. These features can be easily computed from the provided MVs and block partition modes, and cover the size, shape, motion, and texture of traffic scenes. We believe that all of features together can rather mitigate the disadvantage of informational insufficiency. Then we propose a Hierarchical Classification based Regression (HCR) model for vehicle counting, as illustrated in Figure 1. HCR divides the traffic scenes into multiple cases according to the vehicle density (*e.g.* heavy, medium, and light here), and then adopts one well-performed regression model for each of them. Indeed, introducing such classification is for better approximating the broad-variation characteristics of traffic scenes, since it is observed in practice that some density-specific patterns are presented for the scenes involving different vehicle counts. Furthermore, we add one more layer of classifiers for handling the boundary cases produced by the first-layer classifiers. As a result, the large estimation deviation incurred by misclassification can be greatly alleviated.

We evaluated the proposed method on the real highway surveillance videos presenting various traffic scenes. The experimental results show that our method is very competitive compared to the pixel-domain methods, which results in the similar performance (the estimation error of $2 \sim 3$), while possesses lower computation complexity.

2. Related Works

Object counting as one of the typical visual tasks targets to estimate the number of specific objects within a given image [15]. According to the adopted strategy, existing approaches can be roughly divided into three categories: counting by detection, counting by clustering, and counting by regression [17]. Usually, the methods based on detection [9, 10] or clustering [23, 34] need to explicitly segment the objects or track the feature points, and thus may fail if the serious occlusions or scene clutters arise. Differently, the regression based methods [3, 6, 33] are to straightforwardly learn a mapping from the extracted image features to the desired density value. Such a way can alleviate the ubiquitous interferences and thus is usually outperforming in practice along with the advantage of simplicity. Hence the regression model is considered more applicable to counting objects from the realistic scenes [17].

In the previous literatures, most of the proposed regression models are primarily aimed to crowd counting in public [25]. For example, Davies *et al.* [7] first proposed a linear regression model mapping the holistic features into the people count. Chan *et al.* [3] proposed a perspective normalization method to handle the diversity of camera perspectives and a bank of complementary features to improve the accuracy. In addition, some semi-supervised counting methods [27, 31] were also proposed, which were principally to utilize the continuity and consistency between unlabeled samples and their temporally neighboring samples. Comparatively, these methods can exploit more unlabeled data, *e.g.* via transfer learning [18], and thus perform better for the complicated crowd scenes difficult to label. Recently, convolutional neural network (CNN) was specially introduced to cross-scene crowd counting [33]. Specifically, the model was pretrained using a given dataset and then finetuned for an unseen target scene by feeding the retrieved similar training samples.

However, existing regression methods for vehicle counting are very rare, which practically is rather difficult due to the visual diversity of vehicle appearance (*e.g.* truck vs. car). To our best knowledge, only a three-level cascaded regression model in pixel domain [16] was proposed to classify the vehicle scenes, and no any compressed-domain method has been investigated yet. Actually, current works in compressed domain mostly focus on the detection and segmentation of moving objects [13, 26, 14]. Other related works to vehicle counting are to estimate some traffic parameters, *e.g.* the congestion level, and vehicle speed. Specifically, Porikli *et al.* [22] proposed a traffic congestion estimation method by analyzing the MPEG-encoded videos, where the DCT coefficients and MVs were exploited. Tusch *et al.* [29] introduced four features associated with the vehicle density to estimate level of service (LOS). Yu *et al.* [32] proposed to estimate the mean vehicle speed

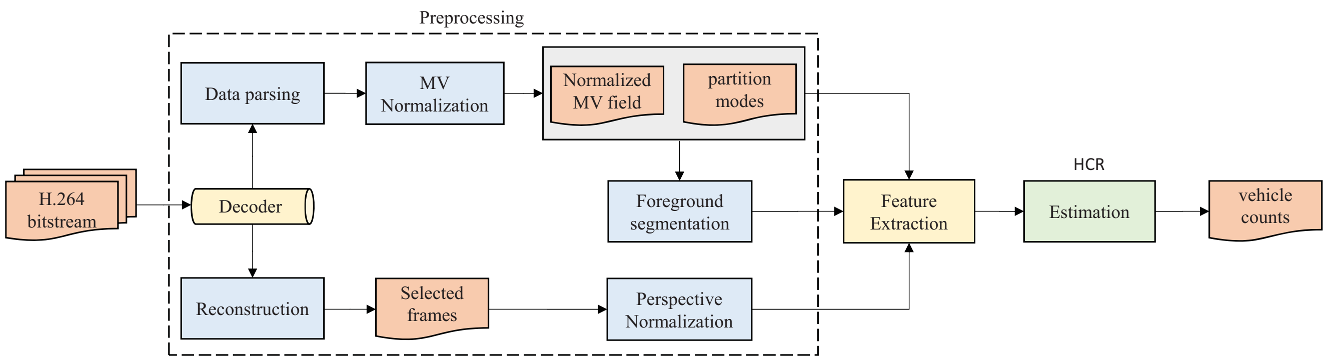

Figure 2. The framework of our proposed highway vehicle counting system, which includes three key stages: (1) video preprocessing, (2) feature extraction, and (3) estimation of vehicle count. The preprocessing stage is to extract the video encoding information from input video bitstream, and then prepares the necessary data for extracting features. Then these data are translated into various features to represent the complex traffic scenes. Finally, the number of vehicles is estimated using the proposed HCR.

using MVs of MPEG-encoded videos.

3. Our Approach

3.1. Overview

In this paper, we propose to solve the vehicle counting issue in compressed domain, aiming at approaching the performance achieved by the pixel-domain methods. The main obstacles to vehicle counting in compressed domain lie in the limited and inaccurate provision of available information and broad-changing of traffic scenes. In this work, we tackle these challenges by constructing rich features, which can effectively exploit the provided data for representing the traffic scenes, and proposing a novel counting model, *i.e.* Hierarchical Classification based Regression (HCR), which adaptively applies the suitable submodel for the given traffic scene according to its presenting characteristics.

Figure 2 shows the framework of the proposed counting method. It can be seen that the processing pipeline mainly comprises of three key stages: (1) video preprocessing, (2) feature extraction, and (3) estimation of vehicle count. 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 extracting features. Then we translate these data into various features, which are expected to be rich enough for accurately representing complex traffic scenes. Finally, we conduct the vehicle counting, *i.e.* estimate the number of vehicles, using the proposed HCR. In this paper, H.264 codec [30] is particularly adopted due to 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.

3.2. Preprocessing

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

3.2.1 Motion vector normalization

In the H.264 compressed format, MB is the basic unit of video encoding [30]. The MBs can be encoded in various block partition modes, such as 16×16, 16×8, 8×16, 8×8, 8×4, 4×8, and 4×4, and each block corresponds to a 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} denotes 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 4×4. In order to obtain a uniform MV field, we split all the blocks into 4×4, *e.g.* one 8×16 would be partitioned into 8 pseudo-MBs with the size of 4×4. Particularly, the MVs of 4×4 pseudo-MBs are straightforwardly assigned using the MV of corresponding parent MB. Additionally, for the intra-coded blocks originally having no MVs, we adopt the Polar Vector Median (PVM) [13] method to compute their MVs. Finally, we obtain a normalized MV field $\tilde{V}$.

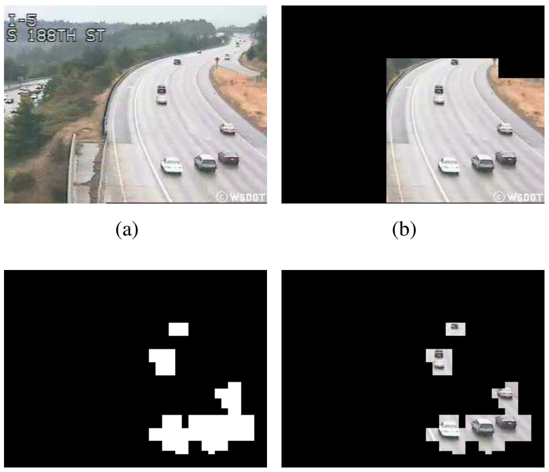

Figure 3. Foreground Segmentation: (a) Original frame, (b) ROI, (c) Foreground mask, and (d) Foreground image.

3.2.2 Macro-block weighting

There are seven different partition sizes in H.264 with the application of deformable macro-block technology. Particularly, the areas around moving objects usually have small partition sizes in order to achieve higher compression efficiency [26]. Therefore, the blocks with smaller partition sizes are more likely to represent the actual vehicle motion, and thus are expected to make more contributions to holistic features. We assign MBs different weights according to the MB partition modes. Let $f_m(B_{ij})$ denotes the partition mode of B_{ij}. Then the weights are determined by following the rules as:

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

where PB_{ij} denotes the parent block of the 4×4 pseudo-block B_{ij}.

3.2.3 Foreground segmentation

This process is to separate the foreground regions from background in the normalized MV field. To this end, we first apply a binary region of interest (ROI) to the MV field. Then we adopt the thresholding strategy to produce the final foreground regions, *i.e.* the block B_i inside the ROI is labeled as foreground if its MV is larger than the preset threshold T_f. To capture the vehicle motion in all scenes, the threshold is set as $T_f = 1$ in our implementation. Figure 3 presents an exemplar of segmentation result. It can be observed that the moving vehicles can be roughly contained by the segmented foreground regions although part of backgrounds may also be involved. In particular, adopting the simple thresholding strategy here is due to its low complexity, and we believe that the performance would be further improved if some advanced methods are taken.

3.2.4 Perspective normalization

In the video surveillance systems, the far vehicles appear smaller than those closer to the camera due to the perspective effects. Consequently, the features extracted from the same object with different scene depths would be diversified. To deal with such an issue, the perspective normalization is usually performed. Practically, each block is scaled 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 [3] 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.

3.3. Feature extraction

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

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 denoted by f_a is calculated from the perspective normalization map S and MB type weights W_{ij}, *i.e.*

$$f_a = \sum_{B_{ij} \in \mathcal{F}} W_{ij} \cdot S_{ij}, \quad (3)$$

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 denoted by f_l is weighted using MB type weights W_{ij} and square root of perspective normalization map S as in [3]:

$$f_l = \sum_{B_{ij} \in \mathcal{P}} W_{ij} \cdot \sqrt{S_{ij}}, \quad (4)$$

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

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:

$$\begin{aligned} o_{ij} &= \tan^{-1}\{g_y(\tilde{V}_{ij})/g_x(\tilde{V}_{ij})\} \\ m_{ij} &= \sqrt{g_x(\tilde{V}_{ij})^2 + g_y(\tilde{V}_{ij})^2} \end{aligned}, \tag{5}$$

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 when computing the histogram, and is $(m_{ij} \cdot W_{ij} \cdot \sqrt{S_{ij}})$.

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

Texture: The texture features are strongly correlated to the vehicle density for traffic scenes. Compared with scenes of low density, scenes of high density tend to present finer patterns [20]. 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) [11, 3] and local binary pattern (LBP) [19]. In this work, we particularly employ the LBP operator [21] due to its simplicity and effectiveness. The LBP feature is constructed by comparing the MVs of eight-neighboring MBs to the target one. Formally, LBP of the target MB B_{ij} is defined as:

$$\mathrm{LBP}_{ij} = \sum_{B_k \in \mathcal{N}_{ij}} s(d(\tilde{V}_{ij}, \tilde{V}_k)) \cdot 2^k, \tag{6}$$

where $\mathcal{N}_{ij}$ represents the set of neighboring MBs of B_{ij}, and B_k is its element with $k = 0, 1, \cdots, 7$. In addition, $d(\cdot)$ is a distance metric function to measure the similarity between two MVs, which is defined as:

$$d(\tilde{V}_i, \tilde{V}_j) = \exp\left\{ -\frac{\|\tilde{V}_i - \tilde{V}_j\|^2}{\|\tilde{V}_i\|^2 + \|\tilde{V}_j\|^2} \right\}. \tag{7}$$

And $s(\cdot)$ is a sign function:

$$s(x) = \begin{cases} 1 & x \geq T_s \\ 0 & x < T_s \end{cases}. \tag{8}$$

Here T_s is a threshold, which is set to 0.9 throughout our experiments.

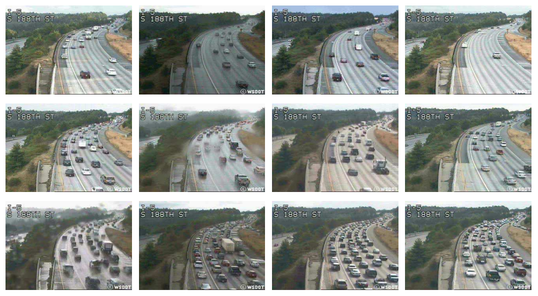

Figure 4. Example frames from the UCSD highway traffic dataset. The sample frames present various vehicle densities: light (top row), medium (middle row) and heavy (bottom row).

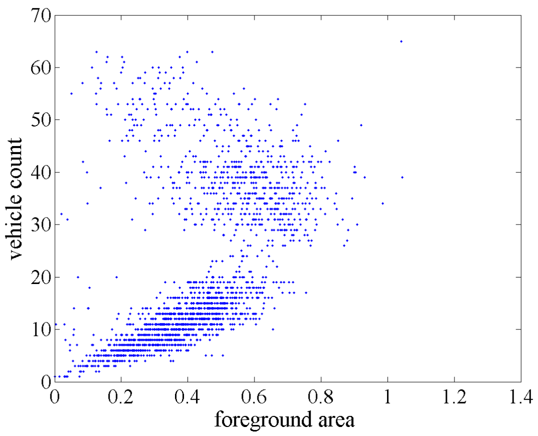

Figure 5. Correspondence between vehicle count and foreground size. The correlation is quite complicated compared to the simple linearity, while the local linearity is approximately held if the vehicle density varies within a small range.

The final texture feature employed 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.

3.4. Counting method

Now we investigate the vehicle counting method that is used to estimate the number of vehicles. In the previous works, the regression based methods have shown impressive performance [17, 25]. Thus we also adopt the regression as the base model. This work is the first 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 [3] or Mall dataset [6], in which the foreground areas vary nearly linear with the number of people. However, such a correlation has not been held yet for vehicle counting, especially in compressed domain.

To intuitively show the characteristics of traffic scenes, Figure 4 provides some realistic highway images, and Figure 5 demonstrates an exemplar of the relationship between the vehicle count and foreground area. It can be seen that the correlation is quite complicated compared to the simple linearity. Theoretically, the major factors causing such complication include the broad variation of vehicle appearance, inaccurate information provision 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. Here it is considered that the local linearity is approximately held if the vehicle density[1] only varies within a small range. That is, the traffic scenes containing different numbers of vehicles present the density-specific patterns. This assumption practically is reasonable according to the results in Figure 5, and exactly turns to be the core idea of HCR.

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 ($K = 3$ is adopted in our experiments with *heavy*, *medium*, and *light* as used in [4]). However, the estimation error may be rather bigger once the input traffic scene is misclassified. Then we also introduce the second-layer classifiers to deal with the boundary cases, which together with the first-layer classifiers form a soft-segmented multiple classifiers. Here the $(K - 1)$ binary classifiers are deployed in the second layer, each of which takes charge of one boundary area. When an new traffic scene arrives, it would be separately classified by both layers of classifiers, and two or three confidence scores are obtained. In particular, only the samples with the scores in the second layer more than a given threshold $T_c = 0.7$ 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 [5] to perform the classification, and Gaussian Process Regression (GPR) [24], which does not impose 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 [3] is employed:

$$\mathcal{K}(\mathbf{x}_i, \mathbf{x}_j) = a_0 + a_1 \mathbf{x}_i^T \mathbf{x}_j + a_2 \exp\left(-\frac{|\mathbf{x}_i - \mathbf{x}_j|^2}{2a_3}\right) + \delta_{ij} a_4. \tag{9}$$

Here δ_{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 term captures local non-linearities, while the last term models the observation noise.

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

4. Experiment

There are three types of temporally interleaved frames in H.264 bitstream [30]: I-frame, P-frame and B-frame. I-frame is absolutely intra-coded, P-frame is motion compensated in the forward direction from I-frame or other P-frame, and B-frame is motion compensated in both forward and backward directions. In our experiments, only 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[2]. We use the same frame features extracted in Section 3.3 for both classification and counting.

4.1. Dataset

Due to the lack of benchmark database for vehicle counting, we adopt the UCSD highway traffic dataset [4] for evaluation. 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 320×240 pixels. Figure 4 provides some exemplar 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).

The UCSD dataset was originally built for classification. To perform the vehicle counting considered in this work, we constructed the corresponding counting dataset with the same traffic videos. Specifically, 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 USCD dataset.

As for the classification, we define the dense categories by counting the labeled vehicles. Specifically, the samples containing vehicles less than 20 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 $[36, 44]$.

4.2. Classification results

In the experiment, the samples were randomly split into the training set containing 800 samples and the test set

[2]`http://iphome.hhi.de/suehring/tml/`

Number of training samples	200	300	400	500	600	700	800
size	3.700	3.411	3.441	3.358	3.145	3.034	2.935
size + shape	3.641	3.360	3.338	3.315	3.009	2.909	2.808
size + shape + HOMV	3.582	3.324	3.288	3.281	2.899	2.824	2.702
Ours (with all features)	3.389	**3.214**	**3.193**	**3.063**	**2.813**	2.736	2.593
[3]	3.324	3.294	3.217	3.074	2.838	2.650	2.653
[25]	**3.301**	3.242	3.196	3.181	3.035	**2.586**	**2.543**

Table 1. Comparison of approaches and feature sets on UCSD dataset with the first splitting strategy.

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
Ours (with all features)	**3.639**	3.483	**3.420**	3.332	**3.142**	2.956	2.938
[3]	4.140	3.581	3.439	3.515	3.229	3.020	2.970
[25]	3.866	**3.443**	3.432	**3.268**	3.170	**2.923**	**2.917**

Table 2. Comparison of approaches and feature sets on UCSD dataset with the second splitting strategy.

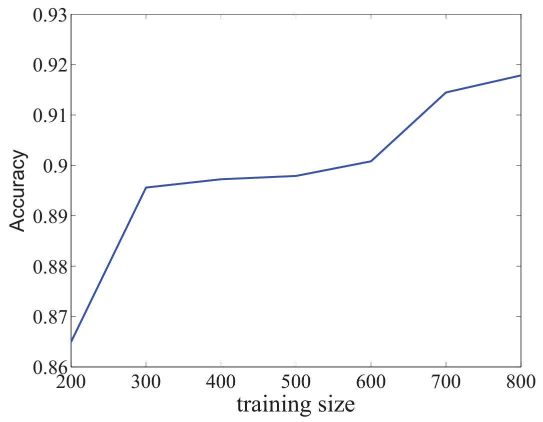

Figure 6. Classification accuracies for different training sizes.

holding the remaining 1232 samples to balance the different traffic patterns and weather conditions in the training and test set. The feature of any frame for classification is generated by averaging the features of five consecutive neighboring frames in order to improve the robustness. We increase the number of training samples from 200 to 800 with the interval of 100 to investigate its influence to the performance. For each training setting, we repeat the experiment five times, and report the mean classification accuracy. Figure 6 gives the classification results under different numbers of training samples. It can be seen that the classification accuracy is around 90% and increases as more training samples are used.

In addition, we compare the proposed method to the pixel-domain baseline methods [4, 8]. For fair comparison, the same training/test samples and settings are used, and the average features on the whole video clips are adopted as video representations. On this dataset, we finally achieve the mean classification accuracy of 94.22%, which is very close to the best performance of 94.50% achieved by [4] and 95.28% by [8]. These results 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.

4.3. 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. In addition, we also compare the proposed method against the pixel-domain regression methods in [3, 25]. For this type of experiments, two splitting strategies are adopted, and each experiemnt is repeated five times. For the first strategy, the samples are randomly split into the training set containing 800 samples and the test set containing 1232 samples as in the classification experiment. Consequently, the different traffic scenarios and weather conditions are roughly balanced for the training and test sets. The second strategy is to randomly select 80 of 254 videos and use the samples in the selected videos as the training set. By selecting videos rather than samples, we can remove the impact that the training and test samples may from the same video.

Table 1 and Table 2 report the resulting MAEs for both splitting strategies. It can be seen that the counting performance is consistently improved as more features are im-

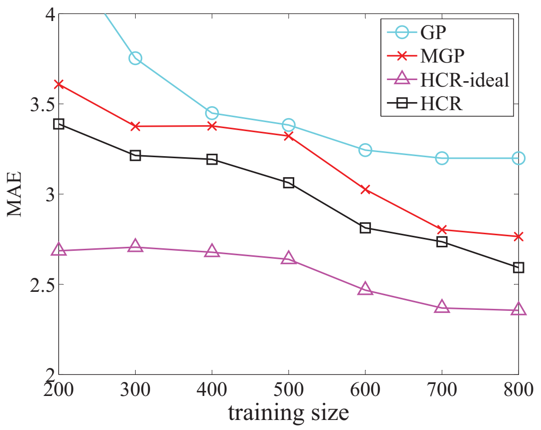

Figure 7. Comparison of MAE for different models. Here GP denotes the single GPR model. MGP represents the one-layer multi-regression model, and HCR-ideal is the ideal version of HCR that adopts the ground truth as classification results.

posed, which demonstrates the effectiveness of all proposed features for vehicle counting. Additionally, our method results in the similar performance to the pixel-domain methods [3, 25], although the second splitting strategy involves a slight performance decrease due to the inconsistence of the sample patterns for training and testing. Thus the proposed method is considered to be very competitive.

We further evaluate the proposed HCR by comparing with the single Gaussian model (GP), one-layer multi-regression model (MGR), and the ideal version of HCR that adopts the classification ground truth to replace the predicted ones. Figure 7 provides an intuitive comparison performance for different methods. Evidently, the multiple regressions always outperform the single regression, and the introduced second-layer classifiers in HCR put the counting performance towards the optimal results.

5. Conclusion

In this work, we present a highway vehicle counting method in compressed domain. Our purpose is to achieve acceptable estimation performance approaching the pixel-domain methods. Specifically, we first developed a batch of low-level features by utilizing the codec metadata of compressed videos. Then we proposed a hierarchical classification based regression model (HCR) to estimate the number of vehicles. Finally, we verified the effectiveness of the proposed method through the experimental evaluation. This work shows that the compressed-domain method may be very applicable for the real-world video surveillance systems, due to the advantages of low complexity, convenient deployment, and competitive performance.

Acknowledgements

This work is supported partially by the National Natural Science Foundation of China under Grant 61233003 and 61203256, Natural Science Foundation of Anhui Province (1408085MF112), and the Fundamental Research Funds for the Central Universities (WK3500000002 and WK3490000001).

References

[1] R. V. Babu, M. Tom, and P. Wadekar. A survey on compressed domain video analysis techniques. *Multimedia Tools and Applications*, 2014.

[2] E. Baş, F. S. Salman, et al. Automatic vehicle counting from video for traffic flow analysis. In *IVS, IEEE*, 2007.

[3] A. B. Chan, Z.-S. J. Liang, and N. Vasconcelos. Privacy preserving crowd monitoring: Counting people without people models or tracking. In *CVPR*, 2008.

[4] A. B. Chan and N. Vasconcelos. Classification and retrieval of traffic video using auto-regressive stochastic processes. In *IVS, IEEE*, 2005.

[5] C.-C. Chang and C.-J. Lin. LIBSVM: A library for support vector machines. *TIST*, 2011.

[6] K. Chen, C. C. Loy, S. Gong, and T. Xiang. Feature mining for localised crowd counting. In *BMVC*, 2012.

[7] A. C. Davies, J. H. Yin, S. Velastin, et al. Crowd monitoring using image processing. *Electronics & Communication Engineering Journal*, 1995.

[8] K. G. Derpanis and R. P. Wildes. Classification of traffic video based on a spatiotemporal orientation analysis. In *WACV*, 2011.

[9] P. Dollar, C. Wojek, B. Schiele, and P. Perona. Pedestrian detection: An evaluation of the state of the art. *TPAMI*, 2012.

[10] P. F. Felzenszwalb, R. B. Girshick, D. McAllester, and D. Ramanan. Object detection with discriminatively trained part-based models. *TPAMI*, 2010.

[11] R. M. Haralick, K. Shanmugam, and I. H. Dinstein. Textural features for image classification. *TSMC*, 1973.

[12] S.-C. Huang and B.-H. Chen. Automatic moving object extraction through a real-world variable-bandwidth network for traffic monitoring systems. *TIE*, 2014.

[13] S. H. Khatoonabadi and I. V. Bajić. Video object tracking in the compressed domain using spatio-temporal markov random fields. *TIP*, 2013.

[14] S. S. Kruthiventi and R. V. Babu. Crowd flow segmentation in compressed domain using crf. In *ICIP*, 2015.

[15] V. Lempitsky and A. Zisserman. Learning to count objects in images. In *NIPS*, 2010.

[16] M. Liang, X. Huang, C.-H. Chen, X. Chen, and A. Tokuta. Counting and classification of highway vehicles by regression analysis. *TITS*, 2015.

[17] C. C. Loy, K. Chen, S. Gong, and T. Xiang. Crowd counting and profiling: Methodology and evaluation. *Modeling, Simulation and Visual Analysis of Crowds, Springer*, 2013.

[18] C. C. Loy, S. Gong, and T. Xiang. From semi-supervised to transfer counting of crowds. In *ICCV*, 2013.

[19] W. Ma, L. Huang, and C. Liu. Advanced local binary pattern descriptors for crowd estimation. In *PACIIA*, 2008.

[20] A. Marana, S. Velastin, L. Costa, and R. Lotufo. Estimation of crowd density using image processing. In *IEE Colloquium Image Processing for Security Applications*, 1997.

[21] T. Ojala, M. Pietikäinen, and T. Mäenpää. Multiresolution gray-scale and rotation invariant texture classification with local binary patterns. *TPAMI*, 2002.

[22] F. Porikli and X. Li. Traffic congestion estimation using hmm models without vehicle tracking. In *IVS, IEEE*, 2004.

[23] V. Rabaud and S. Belongie. Counting crowded moving objects. In *CVPR*, 2006.

[24] C. E. Rasmussen. Gaussian processes for machine learning. 2006.

[25] D. Ryan, S. Denman, S. Sridharan, and C. Fookes. An evaluation of crowd counting methods, features and regression models. *CVIU*, 2015.

[26] H. Sabirin and M. Kim. Moving object detection and tracking using a spatio-temporal graph in h. 264/avc bitstreams for video surveillance. *TMM*, 2012.

[27] B. Tan, J. Zhang, and L. Wang. Semi-supervised elastic net for pedestrian counting. *Pattern Recognition*, 2011.

[28] B. Tian, B. T. Morris, M. Tang, Y. Liu, Y. Yao, C. Gou, D. Shen, and S. Tang. Hierarchical and networked vehicle surveillance in its: A survey. *TITS*, 2015.

[29] R. Tusch, F. Pletzer, A. Krätschmer, L. Böszörmenyi, B. Rinner, T. Mariacher, and M. Harrer. Efficient level of service classification for traffic monitoring in the compressed video domain. In *ICME*, 2012.

[30] T. Wiegand, G. J. Sullivan, G. Bjøntegaard, and A. Luthra. Overview of the h. 264/avc video coding standard. *TCSVT*, 2003.

[31] W. Xia, J. Zhang, and U. Kruger. Semisupervised pedestrian counting with temporal and spatial consistencies. *TITS*, 2015.

[32] X. Yu, P. Xue, L. Duan, and Q. Tian. An algorithm to estimate mean vehicle speed from mpeg skycam video. *Multimedia Tools and Applications*, 2007.

[33] C. Zhang, H. Li, X. Wang, and X. Yang. Cross-scene crowd counting via deep convolutional neural networks. In *CVPR*, 2015.

[34] R. Zhao and X. Wang. Counting vehicles from semantic regions. *TITS*, 2013.