FSE-MV: Compressed Domain Video Information Assisted Hybrid Real-Time Vehicle Speed Estimation

Yangjie Cao[1], Qi Wu[1], Bo Zhang[1(✉)], Zhi Liu[2], and Junfeng Li[1]

[1] Zhengzhou University, Zhengzhou, China
{caoyj,zhangbo2050,lijfimp}@zzu.edu.cn, wq1996@gs.zzu.edu.cn
[2] The University of Electro-Communications, Chofu, Japan
liu@ieee.org

Abstract. Vehicular speed estimation is a vital component in intelligent transportation systems. With the recent development of smart cameras and computer vision technologies, video-based vehicle speed estimations have been widely studied. However, facing the huge volume of pixel-domain information, conventional methods are computationally intensive, and often fail to deliver estimation results in real-time. In this paper, we target the video-based real-time vehicle speed estimation problem. For data volume reduction, we utilize the compressed domain video information and propose a hybrid real-time vehicle speed estimation method termed FSE-MV. FSE-MV first segments vehicles using motion vector (MV) information in the compressed domain. The pixel information of the segmented vehicles is then retrieved through decoding. Feature points of each vehicle are extracted for multi-object matching and pixel domain displacement calculation. The speed of the target vehicle is finally calculated through spatial coordinate transformation. Experiments over the public dataset demonstrate that FSE-MV is able to process 1080p traffic video data in real-time (∼30 frames per second) with a high estimation accuracy (∼93.09%).

Keywords: Speed estimation · ITS · Feature matching · Compressed domain

1 Introduction

Vehicle speed estimation is one of the vital components of intelligent transportation systems (ITS) [1,2]. Effective and accurate real-time speed estimation offers unparalleled capabilities for ITS to subsequently monitor and diminish traffic accidents caused by speeding, thereby fully utilize traffic infrastructures as well as protect commuters' assets and lives.

This work is supported by Collaborative Innovation Major Project of Zhengzhou under Grant 20XTZX06013, the Research Foundation Plan in Higher Education Institutions of Henan Province under Grant 20A520037.

C. T. Calafate et al. (Eds.): MONAMI 2021, LNICST 418, pp. 100–114, 2022.
https://doi.org/10.1007/978-3-030-94763-7_8

Current mainstream vehicle speed estimation systems often involve sensors such as inductive loop detectors, laser meters, and Doppler radars [3]. The inductive loop detectors require complex installation and maintenance. Laser meters and Doppler radars often incur a high cost and suffer from environmental restrictions.

With the advancement of digital cameras in terms of both data quality and cost-effectiveness, as well as the development of computer-vision techniques, both academia and industry shift their attentions to video-based speed estimation [3–9]. It is inevitable for all video-based vehicle speed estimation techniques to first detect the vehicles from captured frames. Vehicle detection is often addressed by either modeling [4,9], background subtraction [5,6,8] or Neural Networks [7,10,11]. Vehicle matching is then performed to calculate the pixel displacement of the vehicle. Finally, the pixel displacement is converted into real-world position displacement for speed calculation. Depending on the camera deployed, the method of spatial coordinate conversion can be divided into monocular-based approaches [4,5] and stereo-based approaches [7–9]. Monocular cameras require additional parameters to obtain the spatial position of objects in the video. A binocular camera, simulating human eyes, can itself retrieve the spatial position of the video object, but doubles the amount of video data compared to the monocular camera.

Video data is generally encoded into the compressed form prior to transmission or storage [12–15]. Traditional video analysis methods need to scan every pixel of each video frame for object detection. The whole video therefore must be fully decoded for pixel information retrieval. For traffic surveillance servers where multiple incoming live video streams are to be processed, decoding and per-pixel analysis often lead to an unbearably long delay. In contrast, compressed domain methods do not require full video decoding by avoiding the pixel-level calculation of the entire frame for every frame of the video [15–20]. However besides the absence of texture information, compressed domain information also often exhibits high encoding noises. We, therefore, consider combining the strengths of both compressed domain and pixel domain information to achieve fast and accurate speed estimation. In this paper, we propose Fast Speed Estimation based on MV (FSE-MV). FSE-MV utilizes compressed domain information for vehicle segmentation, to avoid per-pixel processing. We focus only on the moving part and not the whole frame. Contributions of this work are summarized as follows:

- We propose FSE-MV, a hybrid real-time vehicle speed estimation method with the highly accuracy and efficiency. The method uses compressed domain information to assist the pixel domain to reduce the computational complexity.
- FSE-MV includes an effective denoising method, region of interest (ROI) segmentation and object track base on the compressed domain, feature matching base on compressed and pixel domains, and a commonly coordinate translation.
- This paper tests FSE-MV on real-world surveillance video data, confirming that the method, running on a personal laptop with modest computational capacity, can process 1080p video format and perform estimation at 93.09% accuracy, with the processing speed of up to 30fps.

The remainder of this article is as follows: In Sect. 2, we discuss the related work. In Sect. 3, we introduce FSE-MV in detail. In Sect. 4, we demonstrate and analyze the experiment result. The conclusion is placed in Sect. 5.

2 Related Works

In this section, we will introduce compressed domain segmentation approaches and the pixel-based vehicle speed estimation methods respectively.

2.1 Compressed Domain Target Segmentation

The compressed domain methods require only partial decoding of the sparse cues such as motion vectors (MV), transform coefficients, quantization parameters (QP), macro-block partition modes, etc. [16]. Poppe et al. [21] proposed a background model based on the macro-block (MB) sizes within a frame, predicting that moving objects generally involve more bits in MB than the background. However, the number of encoded bits of complex regions is also higher, affecting the foreground segmentation accuracy. Ma et al. [22] introduced a motion target detection algorithm. The algorithm uses both MV and QP as features. They construct a Markov Random Field (MRF) model for foreground and background differentiation. The system only works when *QP adaptation* is enabled at the encoder. Chen et al. [23] proposed to use an eight-parameter model to compute frame-level global motion and compensate the MV field before performing motion segmentation. Their method shows good accuracy for videos with moving cameras. However, the proposed iterative approach accumulates error over time.

2.2 Vehicle Speed Estimation

Most speed estimation methods involve a background/foreground segmentation step to detect region of interest, such as modeling [4], background subtraction [6,8], and Neural Networks [7].Vehicle speeds are measure by tracking vehicle features. Luvizon et al. [4] proposed to use *Motion History Image* (MHI) to detect vehicles and use the *FIND-HILLS* routine to determine vehicle boundaries. They then used a license plate detector to locate the license plate. Speed is calculated by tracking the features of the license plate. Afifah et al. [6] used background subtraction to determine the vehicle location and form the vehicle contour. They first acquire the complete background by accumulating images without vehicles. Then the background is Gaussian filtered to obtain the foreground mask. Morphological filtering is performed to remove the noise from the foreground mask to form the target profile. Vehicles are then tracked from the distance of centers from one frame to another frame. Yang et al. [7] used the stereo cameras for speed estimation. Their system uses an optimized single shot multi-box detector network that can efficiently detect license plates, which are then used for vehicle matching as well as speed calculation. Bouziady et al. [8]

presented a technique to estimate vehicle speed on the highway using stereo images. They use stereo cameras to capture images and determine the vehicle by subtracting the background. Finally, they use feature point matching to calculate the speed.

Most existing video-based speed estimation methods require locating vehicles in the pixel domain of the entire frame. We instead consider using compressed domain information to reduce the complexity introduced by the pixel domain. MVs exist in a variety of standards with commonality. Therefore, FSE-MV using MV is more versatile. Our analysis in the compressed domain avoids iteration and is confined only in adjacent frames. In this case, the error accumulation is also avoided.

3 FSE-MV: Fast Speed Estimation Based on MV

In this section, we describe the proposed FSE-MV in detail.

Table 1. Major symbols used in this paper

Symbol	Denotation
MV_i	The motion vector of MB_i
$MVx_i \& MVy_i$	The horizontal and vertical coordinate of MV_i
E_i	The energy of MV_i
ηMV	Projection motion vector
H	Homography matrix
α	Feature point removal threshold
S_i	Displacement of a matching pair of points
V	Speed of current frame

3.1 Overview of FSE-MV

The overall process of FSE-MV is summarized in Fig. 1. MVs of a given video are first extracted and pre-processed for target segmentation and ROI determination. ROIs are then tracked across adjacent frames to obtain ROI pairs. Feature points of ROI pairs are extracted and matched. Through coordinate transformation, we map the pixel-level displacement to real-world displacement. The actual speed of objects can then be calculated.

FSE-MV includes the following steps: MV extraction, MV pre-processing, target detection and segmentation, target tracking, and speed estimation with coordinate transformation. We next describe each step in detail. Major symbols used in this work are summarized in Table 1.

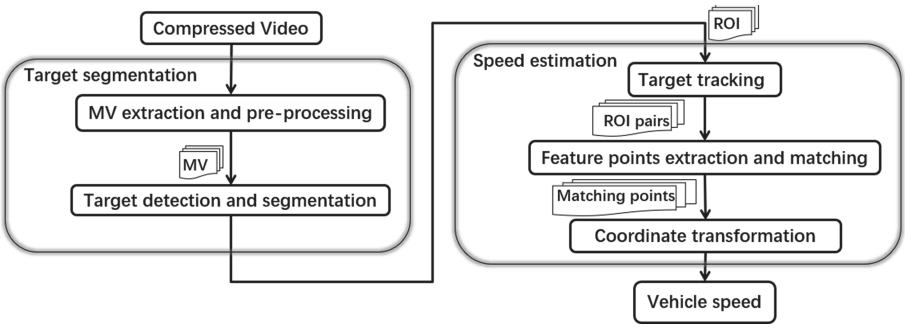
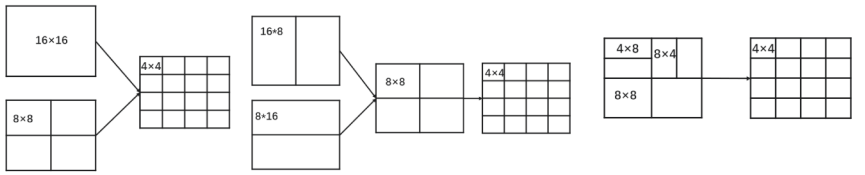


Fig. 1. Overview of FSE-MV.

3.2 MV Extraction



(a) Square (b) Rectangle with long side 16 (c) Rectangle with long side 8

Fig. 2. Example of macro-block normalization

Motion vector (MV) is a two-dimensional vector with a horizontal parameter x and a vertical parameter y in the compressed domain. In the widely applied H.264 coding standard, the smallest coding, macro-block (MB), is assigned with one MV. The sizes of MBs are adaptive from 16×16 through 4×4 pixels. FSE-MV uses FFmpeg [24,25] to extract the MVs, before MB normalization and MV replication. We normalize the sizes of MBs to 4×4. A square MB is directly split into 4×4 blocks as in Fig. 2(a). Otherwise, the MB is first divided into square MBs before splitting, as shown in Fig. 2(b) and 2(c). All MVs of little MBs are given by large MBs. The extraction result is called the MV field.

3.3 MV Pre-processing

MVs are mainly generated for coding efficiency. The MV field is therefore severely affected by noise. We propose to utilize the continuous and smooth property of actual motion for noise removal. The pre-processing of MVs includes three steps: filtering, MV spatial noise removal, and MV temporal noise removal.

Filtering. FSE-MV filters the MV field to remove zero-MV. Zero-MV is the MV with 0 energy. The energy of MV_i is defined as E_i, given by Eq. (1). MBs with zero-MV are skipped in the subsequent processing.

$$E_i = (MVx_i)^2 + (MVy_i)^2. \tag{1}$$

MV Spatial Noise Removal. FSE-MV uses spatial noise removal to eliminate isolated MVs, since they rarely represent actual moving objects. The spatial noise removal process is shown in Algorithm 1. For each MV in the MV field, if for all its surrounding MVs, less than half (50%) are zero-MV, then this MV is deemed useful MV, and is added into the result (lines 2–7).

Algorithm 1. MV Spatial Noise Removal

INPUT: MV field F
OUTPUT: Spatially denoised MV field F'
1: $F' = \emptyset$
2: **for** $f \in F$ **do**
3: S_f is the set of surrounding blocks of f
4: **if** count$(i \in S_f \& E_i = 0) \leq 50\%$ len(S_f) **then**
5: $F' = F' \cup \{f\}$
6: **end if**
7: **end for**
8: **return** F'

MV Temporal Noise Removal. FSE-MV uses Algorithm 2 to remove residual fake MVs through temporal analysis. Each block f is reversely mapped to location b_f of the preceding frame according to its MV_f (lines 3–4). If b_f has non-zero MV, MB f is considered as a useful block and stored in F_t'' (lines 5–6). Otherwise, we map position of f to location b_f in the subsequent frame (lines 8–9). If b_f has non-zero MV, f is considered as a useful block and stored in F_t'' (lines 10–12). The returned MV field is then used for motion analysis.

Algorithm 2. MV Temporal Noise Removal

INPUT: Previous frameF'_{t-1}, Current frame F'_t, Next frame F'_{t+1}
OUTPUT: Temporal denoised MV field F''_t
1: $F''_t = \emptyset$
2: **for** $f \in F'_t$ **do**
3: Calculate the best match block $b_f \in F'_{t-1}$
4: b_f's coordinates =f's coordinates $+MV_f$
5: **if** $E_{b_f} \neq 0$ **then**
6: $F''_t = F''_t \cup \{f\}$
7: **else**
8: Calculate the best match block $b_f \in F'_{t+1}$
9: b_f's coordinates =f's coordinates -MV_f
10: **if** $E_{b_f} \neq 0$ **then**
11: $F''_t = F''_t \cup \{f\}$
12: **end if**
13: **end if**
14: **end for**
15: **return** F''_t

3.4 Target Detection and Segmentation

FSE-MV now determines the region of moving objects in each frame based on these MVs. A binary mask is first constructed to marks the pre-processed MVs (set to white) and isolates vehicles from the background.

FSE-MV performs morphological filtering on the vehicle mask to remove voids or gaps and then checks the connectivity of the white area to form the ROI. FSE-MV obtains pixel domain information of the ROI and analyzes only this information.

3.5 Target Tracking

Algorithm 3. Target Tracking

INPUT: ROI $A(x, y, w, h)$, Next frame ROI M($m_1, m_2...m_i$)
OUTPUT: Tracking target m_t

1: $IOU_{max} = 0$
2: Calculate ηMV by Eq. (3)
3: Move A with ηMV to obtain $A'(x', y', w, h)$
4: $x' = x - \eta MV_x$
5: $y' = y - \eta MV_y$
6: **for** $m_i \in M$ **do**
7: $\quad IOU_{max} = Max\{IOU_{max}, IOU(A', m_i)\}$
8: $\quad$ **if** IOU_{max} is changed **then**
9: $\quad\quad m_t = m_i$
10: $\quad$ **end if**
11: **end for**
12: **if** $IOU_{max} = 0$ **then**
13: $\quad$ **for** $m_i \in M$ **do**
14: $\quad\quad$ Calculate the center-of-mass distance between A' and m_i
15: $\quad\quad$ **if** distance is smaller than current minimum value **then**
16: $\quad\quad\quad m_t = m_i$
17: $\quad\quad$ **end if**
18: $\quad$ **end for**
19: **end if**
20: **return** m_t

Tracking. The ROI of each vehicle needs to be tracked across multiple frames before speed calculation. The tracking is carried out as Algorithm 3. Among the input parameters, (x, y) is the upper-left corner coordinates of the ROI, w and h are ROI's width and height. M contains all ROIs in the subsequent frame. MVs in A are synthesized as the projected MV(ηMV) in lines 2. Then A is moved by ηMV to the next frame to get A' (lines 3–5). A' is then matched with each ROI in this frame by the IOU method (lines 6–11). The one with the largest intersection ratio is used as the tracking object of A (lines 7–9). If there is no intersecting target, the algorithm sets the spatially closest one as the tracking

object (lines 12–19). Let N denote the number of MVs in the ROI, the IOU and ηMV formulas are as follows:

$$IOU(A, B) = \frac{A \bigcap B}{A \bigcup B}, \tag{2}$$

$$\eta MV(x, y) = \frac{\sum_{i=1}^{N} MV_i(x, y)}{N}. \tag{3}$$

Feature Point Matching. FSE-MV obtains the displacement of the same target between frames by feature point matching. We utilize SURF [26], a common feature matching approach, instead of using the ROI center to calculate speed. Compared with other feature point description operators, SURF remains stable in translation and rotation [8,9,27].

The matching is carried out as follows. Feature points are extracted from the ROI. Euclidean distance of feature points is calculated. FSE-MV uses the nearest neighbor distance ratio(NNDR) matching strategy [26] to obtain the best match pairs. Matching points are ranked by the ratio between the shortest and the second shortest distances. If this ratio is less than a preset threshold, they are retained as good matching points. The threshold value is set to 0.7 according to [26]. The tracking is considered failed, if no such matching point is found.

3.6 Speed Estimation with Coordinate Transformation

Coordinate Transformation. By feature matching, we get the matched pixel points. Matching points describe the pixel-level displacement of the vehicle. For real-world speed calculation, FSE-MV requires a coordinate transformation to obtain the actual displacement. FSE-MV uses the homography matrix for coordinate transformation. The homography matrix H may be obtained by associating four points in the image to known coordinates in the real-world plane [4]. Let real-world position coordinates are $(x', y', 1)$ and the pixel coordinates are $(x, y, 1)$, then the conversion equation is:

$$\begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix} = H \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}, H = \begin{bmatrix} h_{11} & h_{12} & h_{13} \\ h_{21} & h_{22} & h_{23} \\ h_{31} & h_{32} & h_{33} \end{bmatrix}. \tag{4}$$

We set $h_{33} = 1$ for H normalization. The other 8 parameters in the matrix H are calculated by solving eight equations through at least two matching point pairs. Matrix H is calculated by marked points in advance.

The coordinates of the matching points belong to the current ROI. The matrix H is for the frame. To unify the coordinate system, coordinates need to be compensated. Let $P_{1i}(x_{1i}, y_{1i})$ be one matching point in ROI. The compensation value (x_0^{ROI}, y_0^{ROI}) is the point of the upper-left corner of the ROI in the frame. The pixel coordinates in the frame are therefore $(x_{1i} + x_0^{ROI}, y_{1i} + y_0^{ROI})$. The real-world coordinates $P_{1r}(x_{1r}, y_{1r})$ then are calculated by Eq. (4).

We get the real-world points of tracking object $p_{1r}(x_{1r}, y_{1r})$ and $p_{2r}(x_{2r}, y_{2r})$ similarly. The real-world displacement can then be calculated by:

$$S = \sqrt{(x_{1r} - x_{2r})^2 + (y_{1r} - y_{2r})^2}. \tag{5}$$

Speed Estimation. The obtained matching points contain noise, i.e., incorrect matches, leading to incorrect displacement calculation. FSE-MV uses trimmed mean to eliminate the effect of incorrect values. We calculate and sort the speeds of all matching points. Then the index of the median of this list is found as *med*. Suppose the number of speeds is *length*. FSE-MV sets a threshold α for extreme value removal. FSE-MV retains the speed values in the interval $[med - \alpha length, med + \alpha length]$. After this, we calculate the instantaneous speed of the vehicle V as follows:

$$V = \frac{\sum_i^N S_i}{Nt}, \tag{6}$$

where S_i is displacements of matching points in current ROI, N is the number of retained displacements, t is obtained from the frames per second.

Our final result should be the average speed of the vehicle over a period of time. So FSE-MV keeps tracking the target vehicle and stops tracking at the stop line, and the average of all estimation speeds is calculated as our result.

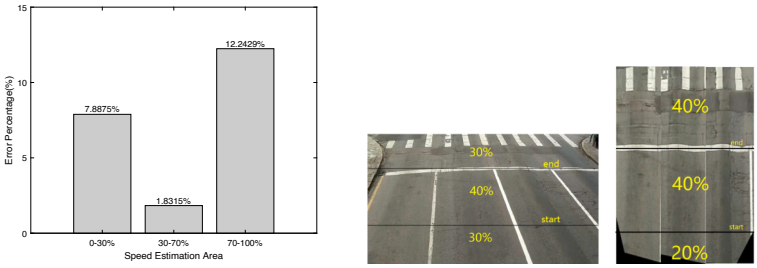
4 Experiment

This section presents the results of our experiments. We first introduce the experiment setup including hardware environment and dataset, followed by a step-wise algorithm demonstration, before discussing quantitative results.

4.1 Setup

Dataset and Environment. The system was tested on full HD quality sequences [4] with the true value of vehicle speed measured by a high precision system of inductive loop detectors. This dataset is annotated with the ground truth displacement. We used two environments for comparison experiments. The CPU environment consists of one 2.8 GHz Intel Core i7 CPU with 8 GB of RAM. The GPU environment includes one RTX 2080Ti graphic card with 12 GB of RAM.

Evaluation Metrics and Comparison Schemes. We evaluate the delay and accuracy of FSE-MV. Delay is defined as the average estimation time required per frame. As for accuracy, an acceptable measurement should be within the $[-3, +2]$ km/h error interval according to [4]. The results lying within this interval are therefore considered to be accurate. For comparison schemes, we substitute *compressed-domain vehicle segmentation* of FSE-MV with YOLOv4 [10] and Fast R-CNN [11] to generate comparing schemes FSE-YOLO and FSE-RCNN. We further compared FSE-MV with Luvizon et al.'s method [4] and Yang et al.'s method [7].



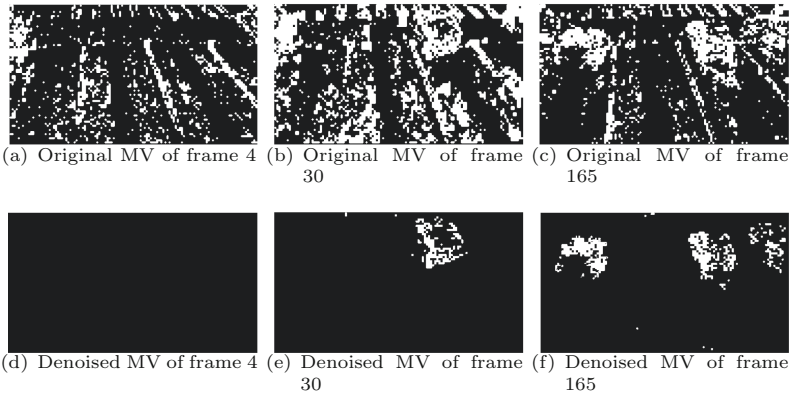
(a) Error distribution of each area (b) SEA in camera (c) Converted SEA

Fig. 3. Set speed estimation area

Speed Estimation Area. Due to the dips of the camera, there may be an area of the captured frame that is more suitable for speed estimation. The optimal speed estimation area (SEA) can be determined by learning. We use the instantaneous speed of the vehicle in different areas to find the mean error of the speed in that area. By continuously dividing the areas, we obtain an area with a minimum mean error for the sequences. As shown in Fig. 3, the average error in the middle part is smaller. As in Fig. 3(c), we converted the original image to show that the scale in the original image is unevenly distributed in reality. This situation is caused by the dips of the camera. This may be one of the reasons why the error in the middle part is smaller. For the sequences used in this paper, we set the bottom of the middle area as the start line and the top as the stop line as in Fig. 3(b).

4.2 Step-Wise Demonstration

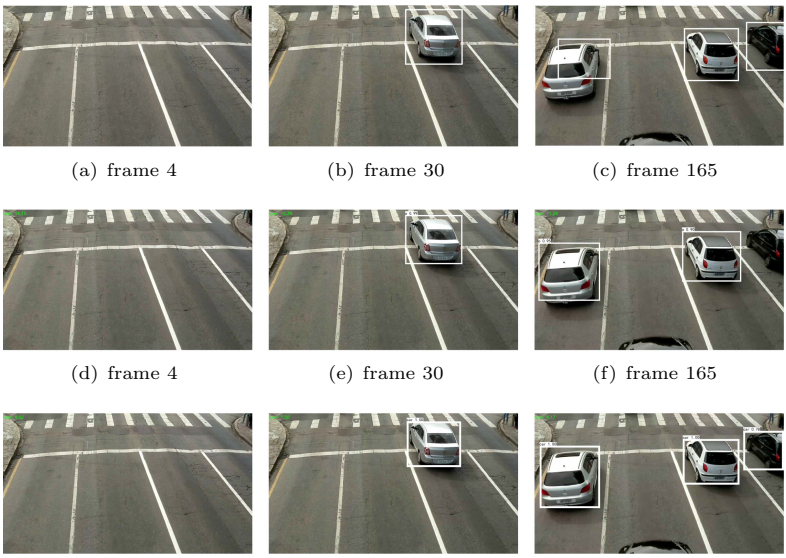
In this section, we demonstrate the per-step workflow of FSE-MV. MV noises can be clearly observed in Fig. 4(a), 4(b), and 4(c), where frame 4, frame 30 and frame 165 represent the case of no-vehicle, single-vehicle, and multi-vehicles respectively. FSE-MV performs spatial and temporal noise removal to obtain clean MV maps as shown in Fig. 4(d), 4(e), and 4(f).



(a) Original MV of frame 4 (b) Original MV of frame 30 (c) Original MV of frame 165

(d) Denoised MV of frame 4 (e) Denoised MV of frame 30 (f) Denoised MV of frame 165

Fig. 4. The effect of MV noise removal



(a) frame 4 (b) frame 30 (c) frame 165

(d) frame 4 (e) frame 30 (f) frame 165

(g) frame 4 (h) frame 30 (i) frame 165

Fig. 5. The effect of vehicle segmentation. (a–c) FSE-MV. (d–f) FSE-YOLO. (g–i) FSE-RCNN

We then segment the vehicle using the MV filed. As can be seen from Fig. 5, the neural network approaches, learning pixel information, result in more reliable segmentation. However, FSE-MV avoids full decoding and thus results in much faster segmentation, as will be discussed later. In our experiment, if its distance is closer than 0.7 times distance of the second nearest neighbor, a matching pair is detected. Matching points are shown in Fig. 6. FSE-MV uses the trimmed mean to remove outliers. We studied the effect of different settings of α. From Fig. 7, it is found that setting 30% has the smallest error. So we set α to 30% for the experiment.

After two ROIs of the same car are matched according to matching points, the pixel-domain coordinates of these points are transformed into real-world coordinates using the homography matrix H. FSE-MV calculates the H for each road lane to facilitate such transform. We are now able to retrieve real-world displacements of these matching points.

Now that we have obtained the real-world displacement together with time duration for matching points, we use Eq. (6) to calculate the vehicle's real-world speed. The evaluation of FSE-MV will be shown in the next section.

4.3 Performance Evaluation

We first analyze the delay of FSE-MV. Figure 8(a) shows the mean delay per frame. The delay is composed of the target segmentation part and the speed estimation part. The values including average delay, minimum delay, and maximum delay are shown in Table 2. FSE-MV is not optimized for GPU, so FSE-MV is not tested

Fig. 6. SURF feature points matching

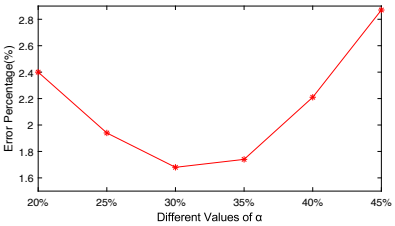
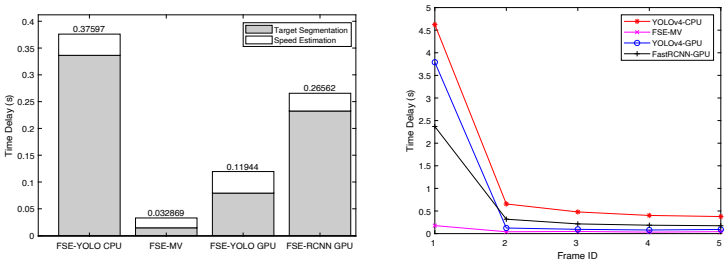


Fig. 7. Estimation error vs. different values of α

in the GPU environment. Since FSE-RCNN has a delay of more than one second in the CPU environment, we choose not to show it. FSE-MV's speed estimation method has the lowest delay. The delay of the speed estimation depends on the area of the ROI. ROI size in Table 2 shows the percentage of ROI against the frame. FSE-MV has a small ROI size and reduces the time delay.



(a) Per-frame processing delay (b) Start-up delay

Fig. 8. Real-time performance

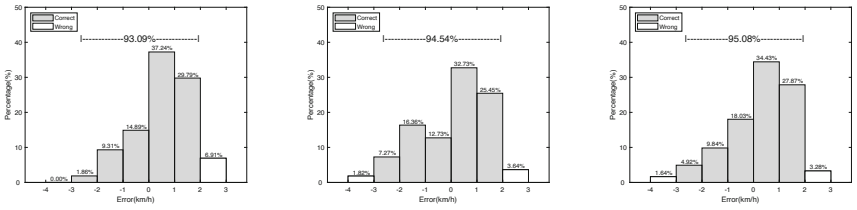
Table 2. Time delay comparison

Method	Target segmentation (s)	Speed estimation (s)	Total (s)			ROI size (%)
			Min	Mean	Max	
FSE-YOLO CPU	0.3364	0.0396	0.3533	0.3760	0.4049	9.7750
FSE-YOLO GPU	0.0793	0.0401	0.0952	0.1194	0.1379	
FSE-RCNN GPU	0.2325	0.0331	0.2417	0.2656	0.3042	8.3578
FSE-MV	0.0145	0.0184	0.0212	0.0329	0.0387	6.8411

We also compared the start-up delay, as shown in Fig. 8(b). Start-up is performed only once, so we compared the start-up delay separately. Neural network based algorithms require model loading and thus need more time for the start-up.

Furthermore, we calculated the speed of the vehicles in the sequences. The results are shown in Fig. 9. FSE-MV is able to reach an accuracy of 93.09%.

The estimation error is mainly introduced by the instability of the feature point matching due to camera tilt. We compared FSE-YOLO and FSE-RCNN under the same condition, results show that they have higher accuracy than FSE-MV. Their target box is more complete, allowing more accurate feature points for matching when calculating the velocity. But FSE-MV still achieves an accuracy rate of over 90%, which is well within the acceptable range.



(a) FSE-MV (b) FSE-YOLO (c) FSE-RCNN

Fig. 9. Speed estimation error distribution

4.4 Experiment Summary

We compared the speed estimation error with other vehicle speed estimation methods, namely, Luvizon et al.'s method [4], Yang et al.'s method [7], FSE-YOLO, and FSE-RCNN, as shown in Table 3. FSE-MV's achieved comparable error with a much lower delay of up to 7-fold, 5-fold, 4-fold, and 8-fold delay reduction comparing to Luvizon et al.'s method, Yang et al.'s method, FSE-YOLO, and FSE-RCNN respectively.

Table 3. Estimation accuracy and delay summary.

Method	RMSE (km/h)	Max error (km/h)	Delay (ms)
Luvizon et al. [4]	1.36	[−4.68,+6.00]	244.8(4.1FPS)
Yang et al. [7]	0.65	[−1.6,+1.1]	185.2(5.4FPS)
FSE-YOLO	0.94	[−3.2,+2.93]	119.4(8.3FPS)
FSE-RCNN	1.25	[−3.69,+2.49]	265.6(3.7FPS)
FSE-MV	1.15	[−2.85,+2.98]	32.9(30.4FPS)

5 Conclusion

In this paper, we studied the video-based vehicle speed estimation method. To address the computational intensiveness of traditional pixel-information based approaches, we propose a hybrid method called FSE-MV that utilizes both pixel domain information and compressed domain information. The system was tested on full HD quality video. Experiments show that FSE-MV is able to achieve an average estimation accuracy of 93.09%, with an overall estimation delay of about 32.9 ms.

References

1. Zhou, P., Chen, X., Liu, Z., Braud, T., Hui, P., Kangasharju, J.: DRLE: decentralized reinforcement learning at the edge for traffic light control in the IoV. IEEE Trans. Intell. Transp. Syst. **22**(4), 2262–2273 (2020)
2. Wu, C., Liu, Z., Liu, F., Yoshinaga, T., Ji, Y., Li, J.: Collaborative learning of communication routes in edge-enabled multi-access vehicular environment. IEEE Trans. Cogn. Commun. Netw. **6**(4), 1155–1165 (2020)
3. Llorca, D.F., Martínez, A.H., Daza, I.G.: Vision-based vehicle speed estimation for its: a survey. arXiv preprint arXiv:2101.06159 (2021)
4. Luvizon, D.C., Nassu, B.T., Minetto, R.: A video-based system for vehicle speed measurement in urban roadways. IEEE Trans. Intell. Transp. Syst. **18**(6), 1393–1404 (2016)
5. Famouri, M., Azimifar, Z., Wong, A.: A novel motion plane-based approach to vehicle speed estimation. IEEE Trans. Intell. Transp. Syst. **20**(4), 1237–1246 (2018)
6. Afifah, F., Nasrin, S., Mukit, A.: Vehicle speed estimation using image processing. J. Adv. Res. Appl. Mech. **48**(1), 9–16 (2019)
7. Yang, L., Li, M., Song, X., Xiong, Z., Hou, C., Qu, B.: Vehicle speed measurement based on binocular stereovision system. IEEE Access **7**, 106628–106641 (2019)
8. El Bouziady, A., Thami, R.O.H., Ghogho, M., Bourja, O., El Fkihi, S.: Vehicle speed estimation using extracted surf features from stereo images. In: 2018 International Conference on Intelligent Systems and Computer Vision (ISCV), pp. 1–6. IEEE (2018)
9. Jiang, J., Mi, C., Wu, M., Zhang, Z., Feng, Y.: Study on a real-time vehicle speed measuring method at highway toll station. In: 2019 International Conference on Sensing and Instrumentation in IoT Era (ISSI), pp. 1–5. IEEE (2019)
10. Bochkovskiy, A., Wang, C.Y., Liao, H.Y.M.: Yolov4: optimal speed and accuracy of object detection. arXiv preprint arXiv:2004.10934 (2020)
11. Girshick, R.: Fast r-CNN. In: Proceedings of the IEEE International Conference on Computer Vision, pp. 1440–1448 (2015)
12. Guo, C., Cui, Y., Liu, Z.: Optimal multicast of tiled 360 VR video. IEEE Wireless Commun. Lett. **8**(1), 145–148 (2018)
13. Liu, Z., Cheung, G., Chakareski, J., Ji, Y.: Multiple description coding and recovery of free viewpoint video for wireless multi-path streaming. IEEE J. Sel. Top. Sign. Process. **9**(1), 151–164 (2014)
14. Zhou, H., Wang, X., Liu, Z., Ji, Y., Yamada, S.: Resource allocation for svc streaming over cooperative vehicular networks. IEEE Trans. Veh. Technol. **67**(9), 7924–7936 (2018)
15. Liu, Z., et al.: Point cloud video streaming in 5G systems and beyond: challenges and solutions. In: IEEE Network (2021)
16. Babu, R.V., Tom, M., Wadekar, P.: A survey on compressed domain video analysis techniques. Multimedia Tools Appl. **75**(2), 1043–1078 (2014). https://doi.org/10.1007/s11042-014-2345-z
17. Jaballah, S., Larabi, M.C.: Fast object detection in H264/AVC and HEVC compressed domains for video surveillance. In: 2019 8th European Workshop on Visual Information Processing (EUVIP), pp. 123–128. IEEE (2019)
18. Zhao, L., He, Z., Cao, W., Zhao, D.: Real-time moving object segmentation and classification from HEVC compressed surveillance video. IEEE Trans. Circuits Syst. Video Technol. **28**(6), 1346–1357 (2016)

19. Zhang, B., Liu, Z., Chan, S.H.G., Cheung, G.: Collaborative wireless freeview video streaming with network coding. IEEE Trans. Multimedia **18**(3), 521–536 (2016)
20. Liu, Z., Zhan, C., Cui, Y., Wu, C., Hu, H.: Robust edge computing in UAV systems via scalable computing and cooperative computing. IEEE Wireless Commun. **28**(5), 36–42 (2021)
21. Poppe, C., De Bruyne, S., Paridaens, T., Lambert, P., Van de Walle, R.: Moving object detection in the H 264/AVC compressed domain for video surveillance applications. J. Vis. Commun. Image Represent. **20**(6), 428–437 (2009)
22. Ma, M., Song, H.: Effective moving object detection in H 264/AVC compressed domain for video surveillance. Multimedia Tools Appl. **78**(24), 35195–35209 (2019)
23. Chen, Y.M., Bajic, I.V.: A joint approach to global motion estimation and motion segmentation from a coarsely sampled motion vector field. IEEE Trans. Circuits Syst. Video Technol. **21**(9), 1316–1328 (2011)
24. Tomar, S.: Converting video formats with FFmpeg. Linux J. **2006**(146), 10 (2006)
25. FFmpeg Developers: FFmpeg tool (Version 4.4) [Software] (2021). http://ffmpeg.org/
26. Bay, H., Tuytelaars, T., Van Gool, L.: SURF: speeded up robust features. In: Leonardis, A., Bischof, H., Pinz, A. (eds.) ECCV 2006. LNCS, vol. 3951, pp. 404–417. Springer, Heidelberg (2006). https://doi.org/10.1007/11744023_32
27. Zhu, Y., Cheng, S., Stanković, V., Stanković, L.: Image registration using BP-sift. J. Vis. Commun. Image Represent. **24**(4), 448–457 (2013)