APSIPA Transactions on Signal and Information Processing, 2023, 12, e106

Original Paper

Real-time Vehicle Detection and Tracking on Fisheye Traffic Monitoring Video in Compressed Domain

Sandy Ardianto[1], Hsueh-Ming Hang[2*] and Wen-Huang Cheng[3]

[1] *Electrical Engineering and Computer Science International Graduate Program, National Yang Ming Chiao Tung University, Hsinchu, Taiwan*
[2] *Institute of Electronics, National Yang Ming Chiao Tung University, Hsinchu, Taiwan*
[3] *Department of Computer Science and Information Engineering, National Taiwan University, Taipei, Taiwan*

ABSTRACT

Our goal is to develop real-time vehicle detection and tracking schemes for fisheye traffic monitoring video using the temporal information in the compressed domain without decoding the entire video. Two algorithms are proposed. The first algorithm starts with a conventional single-frame detector, but we introduce a multi-frame information fusion stage to improve the final detection and tracking accuracy, which is implemented using multi-modal bi-directional LSTM (MM bi-LSTM) network. The second algorithm first constructs multi-frame motion trail image, and then a single-image multi-head detector is designed to produce bounding boxes of an individual frame. The first scheme can be viewed as a detect-to-track design, and the second scheme is track-to-detect. We tested our proposals on the ICIP2020 VIP Cup dataset in H.265 video format. The aforementioned algorithms are applied to the motion fields and residual images in the H.265 compressed data set. It turns out that their detection and tracking performances are on par with their pixel-domain counterparts, and they can achieve the state-of-the-art accuracy of

*Corresponding author: Hsueh-Ming Hang, hmhang@nctu.edu.tw.

Received 25 March 2023; Revised 12 August 2023
ISSN 2048-7703; DOI 10.1561/116.00000116

conventional video object detectors and trackers. If the decoding process for video compression is not counted, their computational complexities are much lower than the conventional pixel-domain video object detectors and trackers.

Keywords: real-time, vehicle detection, vehicle tracking, fisheye camera, compressed domain.

1 Introduction

The proliferation of fisheye cameras in monitoring road intersections is hampered by lens distortion, which enlarges and distorts objects near the image center while compressing those at the periphery. Consequently, conventional object detectors like YOLOv5 [16] suffer from decreased accuracy at the image border. This paper proposes an integrated detection and tracking approach employing deep neural networks, where tracking vehicles across frames enhances detection accuracy, even in the periphery. This synergy leads to mutual reinforcement, wherein effective tracking bolsters detection and vice versa.

In traffic surveillance, videos are typically stored in compressed formats like AVI, AVC, or HEVC to conserve storage. Our study utilizes the H.265 coding standard wherein video sequences are divided into non-overlapping segments termed GOPs, with the first frame being intra-coded (I-frame) and the subsequent frames being predictive (P/B-frame). We exploit the inherent motion vectors and residual information within P/B-frames to expedite the detection process and enhance accuracy without decoding the entire video. Contrary to the common notion that compressed-domain detection and tracking are less accurate than pixel-domain methods, our algorithms demonstrate comparable or superior performance.

We devised two compressed domain algorithms grounded on the integrated detection and tracking principle. The first, a detect-to-track design, initiates with single-frame object detection, whereupon the bounding boxes and cropped images are processed by a multi-modal bi-directional LSTM (MM bi-LSTM) network for simultaneous refinement in detection and tracking. The second, a track-to-detect design, begins by creating a motion trail image from overlapping motion fields and residual images. Simple object detectors identify trails, followed by a multi-head detector, which extracts bounding boxes in individual frames. An optional tertiary stage with MM bi-LSTM can further boost detection accuracy at the expense of additional computation.

Our contributions in this paper can be summarized as follows.

1. We design a detect-to-track algorithm to detect and track vehicles in the fisheye traffic videos. We employ an MM bi-LSTM network to integrate

multi-frame detected bounding boxes and motion field/residual images to improve detection and tracking accuracy.

2. We propose the "motion trail" representation in the compressed domain. It is constructed by overlapping the motion field and residual image in multiple frames. Although individual frame detection has high error rates, the trail image that aggregates multi-frame information can be reliably detected.

3. We extend the pixel-domain track-to-detect algorithm [4] to the compressed-domain. That is, the motion trail in the compressed domain replaces the original frame differences in the pixel domain, and the detection and tracking performance is about the same or higher.

This paper is structured as follows: Section 2 surveys relevant literature; Section 3 outlines the dataset and its attributes; Section 4 details the first compressed-domain detection and tracking algorithm, accompanied by experimental results; Section 5 introduces the second algorithm and its experiments; Section 6 contrasts pixel-domain and compression-domain approaches; and Section 7 offers conclusions.

2 Related Works

2.1 Image Object Detection

Object detector identifies instances of a specific class of objects inside a picture. It is a fundamental component in many computer vision applications. Due to the recent advancement in deep neural networks, many high-performance image object detectors are now available. One-stage methods and two-stage methods are the two primary classifications of contemporary techniques. One-stage approaches address the inference speed, such as YOLO [26], SSD [24], and RetinaNet [22]. Two-stage approaches prioritize detection precision, such as Faster R-CNN [27], Mask R-CNN [14], and Cascade R-CNN [8]. These schemes generate horizontal/vertical bounding boxes (bboxes). For specific applications, the objects can be more accurately bounded by the oriented (rotated) bbox. This is particularly true for, say, the remote sensing images that contain, for example, the image of ships or airplanes on the ground. This is also true for our traffic monitoring cameras, which have a bird's eye view. Several oriented bbox detection papers have been published [28, 44, 45]. The recent state-of-the-art R3Det method [39] showed good performance on DOTA (A Large-Scale Dataset for Object Detection in Aerial Images) dataset [38]. Therefore, it is adopted and modified as a module in our system.

2.2 Video Object Detection

For image sequences containing multiple frames, video object detectors that aggregate multiple frame information to improve detection accuracy have been developed. Some well-known schemes are Detect-and-Track [11], SELSA (SEquence Level Semantics Aggregation) [37], MEGA (Memory Enhanced Global-local Aggregation) [9], and temporal ROI align [13]. The video object detectors extract temporal information often using the techniques of optical flow, attention mechanism, tracking, etc. Therefore, their systems typically need a huge amount of computation and a very large model size. Because our focus is on rigid-body cars and the cars have a rather simple movement on fixed roads (fixed routes), we can design a much faster system that achieves comparable accuracy.

2.3 Tracking and Detection

Object tracking is another popular computer vision research topic. A few popular algorithms are Kalman filter [17] [19], SORT [6], DeepSORT [35], FairMOT [47], and ByteTrack [46]. Typically, they assume objects are well detected by single-frame detectors, and then a tracking method is designed to track the bboxes of the same object over several frames. On the other hand, there are joint detection and tracking methods. For example, Feichtenhofer *et al.* [11] explicitly used the joint detection and tracking concept in algorithm design, and H. Kieritz *et al.* [18] used SSD and RNN to update the tracklet information. Also, FairMOT [47] used the encoder-decoder structure to directly detect and extract the tracklet's features. We adopted the concepts of detect-to-track and track-to-detect in designing our schemes.

2.4 Motion History Image

Because the camera in the traffic monitoring video is stationary, the motion field or residual image in the compressed bitstream provides the moving object information. We thus can use the motion field and residual image to form a trail image. A similar concept called "motion history image (MHI)" was suggested by Bobick and Davis [7]. Because their application is mostly action recognition [1], typically, there is only one object (person) in the picture and its video has no severe lens distortion nor multiple object occlusion. That is, the object can be well detected using simple techniques such as background subtraction, and then the extracted moving pixels of objects are used for constructing the MHI. In our case, the single-frame object detection error rate is over 40% for nighttime videos. Our goal is to improve the detection accuracy by collecting multiple frame information. The motion trail image may contain a high percentage of single-frame errors, but the entire trail made

of multiple frames can still be reliably detected. This principle is similar to that of error-correcting codes. The one-bit error can be corrected in a 3-bit repetitive code. Therefore, to differentiate its image characteristics and its use, our multi-frame motion representation is named "motion trail".

2.5 Compressed Domain Object Detection and Tracking

Most object detection and tracking algorithms were developed to work on the RGB images in the pixel domain. Examples are YOLOv5 [16] and ByteTrack [46]. Recently, a few schemes have been proposed to perform detection and tracking tasks in the compressed domain. Deguerre *et al.* [10] proposed an object detection scheme on a single compressed JPEG image by only partially decoding the part of an image with high-density DCT blocks. It improves the detection speed twice compared to using the same SSD [24] network to detect the object in a fully decoded image. Liu *et al.* [23] used the compressed domain data to develop a real-time person-tracking scheme. They conduct detection only on the keyframe (I-frame) and construct the tracklets in the intermediate frames (P/B-frame) using the motion vector field and residual image without decoding the whole video. Their approach has a good accuracy performance, only slightly lower than DeepSort, but its inference speed is close to a traditional multi-object tracking method. Wang *et al.* proposed a scheme called Motion-aided Memory Network (MMNet) [33], which detects objects using the motion information in the compressed bitstream to speed up P-frame processing. They applied the complete detector on the I-frames. The detected object information propagates to P-frames with the aid of motion vectors. They showed results on test videos containing few objects (less occlusion). Their mAP performance is somewhat lower than the best deep-learning methods at that time, but its speed can be twice as fast or even faster if a lower accuracy configuration is in use. In contrast, we detect objects on the P/B-frames and propagate the detected information to the I-frames. Therefore, we do not decode the I-frames.

3 Dataset and Metrics

In this section, we describe our dataset and its characteristics from the detection and tracking viewpoint. We also describe how the compressed data are pre-processed in order to feed into our proposed schemes. The evaluation metrics are also briefly specified.

3.1 ICIP 2020 VIP Cup Dataset

Only a few open public fisheye traffic monitoring datasets with ground truth are available; the rareness may be due to the reason that they may contain

personal information. The ICIP 2020 Video Image Processing (VIP) Cup dataset [43] was used in this study. The video was captured by a fisheye camera mounted about 8 meters above the ground on several road intersections in a Turkey city. It contains 26 daytime videos and five nighttime videos with a resolution of 1024x1024. Among 31 total videos, 26 (including three nighttime videos) are used for training, and 5 (including two nighttime videos) for testing. Most training videos are around 1,000 frames each, but the nighttime videos are 2,000 to 4,000 frames long. The test videos have about 500 to 600 frames each.

The ICIP20 VIP Cup images were initially stored in JPEG format. We encoded the JPEG-decoded images into the H.265 (High-Efficiency Video Coding (HEVC)) video format using the FFMPEG software [29]. In the H.265 format [34], an image sequence is partitioned into many non-overlapping segments called Group-of Pictures (GOP). The first frame inside a GOP is intra-coded (I-frame, single-frame image coding), and the rest are inter-coded in the P/B-frame format (predictive coding). The P/B-frame data includes the motion field (motion vectors) and the residual image (motion-compensated errors). There are two motion vector lists in H.265, list 0 (L0), which came from the previous reference picture, and list 1 (L1) from the next reference picture. We combine them by summing and merging them into a single channel input for the P/B-frame detection network.

The original motion vector contains two components: a horizontal component and a vertical component. We convert a vector to a scalar by simply adding the values of the horizontal component and vertical component together since the purpose is object detection. Thus, the motion field data used in our system has only one channel. The residual image originally contained three components (RGB), but we converted it into a grey-level image (Value in HSV color space), which has only one channel. Samples of the motion field and residual image are shown in Figure 1. Also shown in Figure 1 are motion trails constructed using motion field and residual image. The motion trails are quantized grey-level images. The colors in the figure are for illustration only; they are used to show different grey levels. Three Group-of Pictures (GOP) sizes, 8, 16, and 32, and several coding bit rates are tested in the experiments. More details will be provided in Sections 4.4 and 5.2, Ablation Study.

The ICIP20 VIP Cup competition organizers provided only the regular bounding box (bbox) for the ground truth. We added the tracking information and the moving status of the cars [3]. In addition, we added the rotated bbox into the ground truth to detect the cars more precisely [4]. Not all vehicles are included in the ground truth of the ICIP 2020 VIP Cup dataset. The stationary cars that are not moving for the entire video are not counted. But if a car moves for a period of time, it is counted as a moving car in the ground truth, even during its stationary periods. For example, if a car starts moving at frame 50, this car is counted as a moving car before frame 50. Thus, the

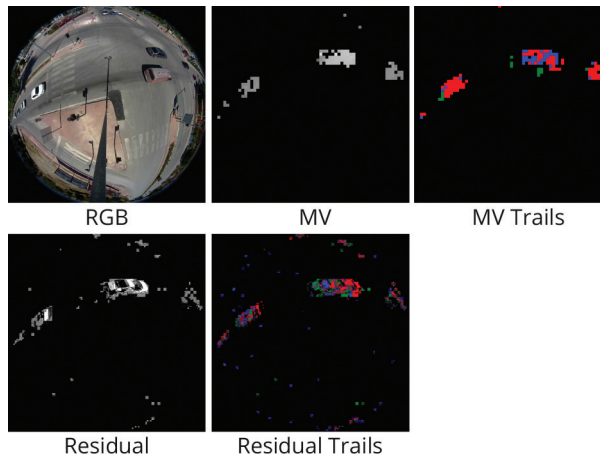


Figure 1: Samples of motion vectors (MV), residual images, and their motion trails. Motion trails are quantized grey-level images. The colors are used to show different grey levels for illustration purposes.

stationary cars are removed in the performance evaluation to make it fair to all object detection methods.

3.2 Dataset Statistics

Figure 2 is cited from [3]. Initially, the dataset has ground truth only for detection (green boxes in Figure 2). We manually added car ID and non-moving cars (red boxes in Figure 2) for tracking purposes. We divide the fisheye image into three rings based on the CKMeans clustering algorithm [32]. The first *inner circle* starts from the center with a radius of 180, the second *middle ring* with a radius from 180 to 320, and the third *outer ring* with a radius from 320 to 512. Based on the statistics in this figure, most cars are located in the outer ring. The rotated bbox is more precise than the normal bbox in representing the vehicle on this top-view fisheye image. The average bbox sizes are more evenly distributed in the inner circle (yellow), which is easier to detect. The number of vehicles using the rotated bbox also spread more evenly in the middle ring than in the regular bbox. Therefore, there is an increase in detection accuracy when using a rotated bbox, as shown in Table 1.

It is worth noting that nighttime and daytime videos have very different image characteristics. This can be seen in Table 1. When we apply YOLOv5 [16] and Rotated YOLOv5 [45] to the ICIP20 VIP Cup dataset, the nighttime video detection rate is much lower. This is due to strong disturbances such as camera noise, car headlight, background light, fading color, etc., in the

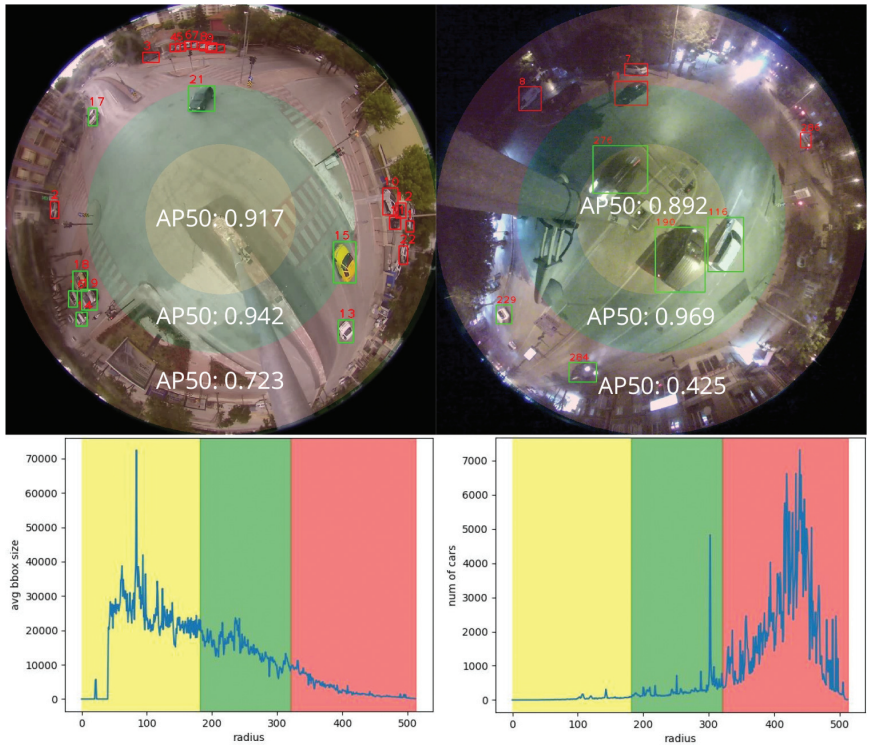


Figure 2: ICIP 2020 VIP Cup dataset samples and statistics. (top left) daytime scene, (top right) nighttime scene, (middle left) average normal bounding box size at a given radius, (middle right) number of cars located at a given radius using a normal bounding box, (bottom left) average rotated bounding box size at a given radius, (bottom right) number of cars located at a given radius using the rotated bounding box [3].

night videos. For example, the camera grain noise is higher under low light conditions. The car headlight can be a clue for detecting a car, but the light from the environment (buildings) may be mixed with it and the car headlight may be occluded by the car in front of it.

The nighttime video characteristics also explain why the rotated bbox produces a more significant improvement (4.6 pp in Table 1) over the regular bbox on the nighttime video. We closely examined the cases where the rotated detection model succeeded while the regular detection model failed. We observed that the headlights of a vehicle in nighttime videos tend to contribute to an inflated bbox when using regular box detection. This enlargement of the bbox causes the Intersection over Union (IoU) score to fall below 50% threshold, leading to a detection failure. An example is shown in Figure 3; the regular detector fails to detect two cars close together, while the rotated object

Table 1: YOLOv5 vs Rotated YOLOv5 detection accuracy on the ICIP20 VIP datasets.

AP_{50}	**Night**	**Day**	**Both**
YOLOv5	0.541	0.820	0.772
Rotated YOLOv5	0.587	0.839	0.786

Figure 3: The normal box detector (left) fails to get the precise vehicle bounding boxes to separate two cars, but the rotated detector (right) is able to construct two tighter bounding boxes. Green box: ground truth. Red box: predicted vehicle location.

detector successfully detects both of them. Since the rotated bbox can adjust its orientation to fit around the vehicle tightly, mitigating the issue caused by various disturbances and thus maintaining an IoU above the 50% threshold.

3.3 Accuracy Metrics

Several accuracy measurements are used in this study, such as AP_{50} (Average Precision) for object detection, rotated AP_{50} for oriented object detection, MOTA (Multiple Object Tracking Accuracy), and MOTP (Multiple Object Tracking Precision) [20] for multi-object tracking. The subscript 50 in AP measurement means that the minimum IoU (Intersection over Union) between the bbox prediction and ground truth is larger than 50%. AP_{50} is suitable for our case since the traffic monitoring task does not need to be very precise as long as the vehicle is detected. Figure 4 shows the IoU calculation for the normal bbox (blue) and rotated bbox (green).

For tracking, MOTA and MOTP are the most popular metrics since they were introduced in the MOT (Multi-Object Tracking) Challenge in 2015 [20].

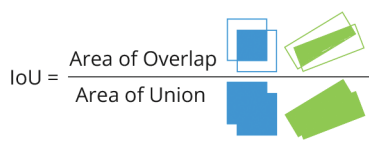


Figure 4: IoU calculation.

Equation (1) specifies the MOTA calculation and Equation (2) for MOTP.

$$MOTA = 1 - \frac{\sum_t (m_t + fp_t + mme_t)}{\sum_t g_t} \tag{1}$$

where m_t is the number of miss detections, fp_t is the number of false positives, mme_t is the number of mismatches, and g_t stands for the number of the ground truth.

$$MOTP = 1 - \frac{\sum_{i,t} d_t^i}{\sum_t c_t} \tag{2}$$

where d_t^i is the distance between the locations of objects in the ground truth and the detection output, and c_t is the total matches made between the ground truth and the detection output. It is the average of the overall predicted location error for all matched object-hypothesis pairings across all frames.

4 Scheme 1: Multi-frame Detection and Tracking

The first Multi-Frame Detection and Tracking (MFD&T) scheme in the compressed domain is described in this section. Because the ICIP20 VIP Cup performance metrics count only the non-stationary vehicles (including the will-move vehicles), we first process the P/B-frames and then extend the extracted information to the I-frames. Because the motion fields and residual images generated by FFmpeg are quite accurate, we can use a relatively lightweight neural net to detect the objects.

4.1 Proposed System

Figure 5 shows the data processing flow in our proposed system. Our system contains two main stages. The first stage is detecting the objects in the motion field and/or the residual image using a simplified rotated bounding box (bbox) detector. The input sequence is divided into overlapped segments. The last frame of the previous segment overlaps with the first frame of the current segment. The overlapped frame is called Key Frame (KF), and the other frames in a segment are Intermediate Frames (IFs). We use R3Det [39] with KFIoU loss [42] to detect the car location because of its good performance in the oriented object detection task. The system architecture is shown in Figure 6. The segment in this example contains 3 frames: two KFs and one IF. Because the inputs data have two channels (rather than 3 channels for the original RGB images) and the image contents are rather simple (large dark background as shown in Figure 1), we found that a smaller backbone network such as ResNet18 for R3Det can produce satisfactory results.

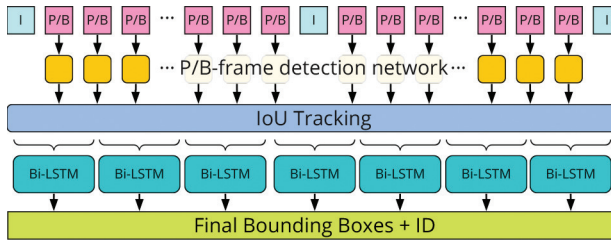


Figure 5: The data processing flow of MFD&T.

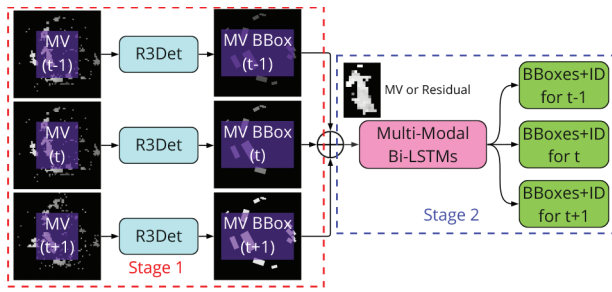


Figure 6: MFD&T system architecture.

The outputs of Stage 1 are the detected bounding boxes (bboxes). We assigned ID numbers to the boxes of the first KF, and the ID numbers were propagated between two nearby frames using a simple maximum IOU criterion. The detected bbox information, together with ID and some clipped input images are fed into the second stage, which refines the labeled bboxes in a segment using a multi-modal bi-directional LSTM (MM bi-LSTM) network. It is also used to fill in the missing bboxes in the I-frames because the I-frames are not processed in the first stage.

The structure of our MM bi-LSTM is shown in Figure 7, which was inspired by Wang [30] [31]. In their original design, they combine text and image information using bidirectional LSTM for the purpose of image captioning. Their neural network stacks two bi-LSTMs [12]. We replace the Text-LSTM with Box-LSTM, which contains the information of rotated bboxes and ID from the outputs of the first stage. The black arrows in Figure 7 are the forward feed, and the red arrows are the backward feed. The yellow circle is the LSTM unit containing 1000 hidden units. The Image-LSTM has the same structure as Box-LSTM. The image inputs to Image-LSTM can be motion field and/or residual image. We assume the maximum number of cars is 100 and each input image size is 64x64 pixels. The top input image is extracted using the detected bbox on the first frame of this segment (first KF), and the

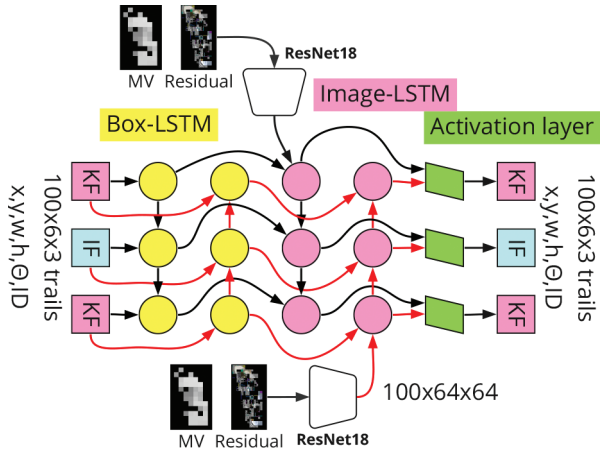


Figure 7: Our multi-modal bi-LSTM architecture for the MFD&T scheme.

bottom image comes from the last frame of this segment (second KF). The extracted image size is normalized to 64x64 before feeding to the Image-LSTM.

The tracking process is easy in our system. Because the KF is shared by two neighboring segments. Once an object ID is assigned in the previous segment, the car ID is passed to the next segment. Therefore, if an object is tracked inside a segment, it is tracked for the entire car path in a video. Because the car speed is relatively slow in passing a road cross, the car bboxes in two neighboring frames typically overlap considerably. Therefore, a simple IOU criterion produces quite promising results. Essentially, this approach is a multi-frame detection and tracking method in the compressed domain; thus, it is abbreviated as "MFD&T".

4.2 Implementation and Training

The first stage of our system uses the R3Det detector [30] with KFIoU loss [12]. We modified the original R3Det architecture for two cases. Because our image (motion field/residual) can be one channel or two channels, we modify the original 3-channel input port of the first convolution layer to one or two channels, but the other parameters of that layer are not changed. In addition, we simplify the backbone network of the original R3Det from ResNet50 to ResNet18 because our motion field or residuals have simpler contents and are easier to detect. Our experiment (Section 4.3) indicates this simplification has little impact on performance.

We start with the pretrained R3Det model. The initial model weights of ResNet18 were pretrained using the ImageNet1K dataset. Then, the R3Det model is retrained using the VIP CUP dataset. We train our neural net using

all the training data, including the daytime and nighttime videos. In the inference (test) phase, we have three testing scenarios: (a) nighttime videos (Night), (b) daytime videos (Day), and (c) both nighttime and daytime videos (Both). In the following experiments, the same single model is used for all three testing scenarios. We also tried to train dedicated models, for example, a "Night model" is trained using the nighttime training data only. However, this dedicated Night model has a slightly lower RD performance on the Night test dataset when compared to the single model trained with all training data. The segment size used in this and the following experiments is 3 unless otherwise stated. Segment size 3 offers the best computing speed; however, the accuracy is slightly lower (0.3 to 0.5 percentage points).

We made two modifications to the second stage bi-LSTM network. Because the inputs to the Image-LSTM port are motion fields/residuals, we modify the original 3-channel input to one-channel or two-channel input. Another major change is that we bypass the first ResNet18 layer of the input port of Image-LSTM and feed the image features directly to the LSTM nodes when the image features are available. Since we use the same backbone, ResNet18, in R3Det, we store the image features generated by R3Det at its Refinement Stage output and feed them directly to the second stage of the Image-LSTM port to save computation. In the implementation, the image features are stored using PyTorch's forward hook, then crop and scale the features to 64x64. Therefore, the speed of this modified MM bi-LSTM is comparable with the box input-only LSTM network. The bi-LSTM has no pretrained model. It is trained using the data generated by the first stage of the proposed scheme.

We train our schemes on four NVIDIA 1080Ti GPUs and an i9-9900x CPU using PyTorch 1.7.0. The hyper-parameters are as follows: learning rate with an initial value of 0.0025, momentum of 0.9, and weight decay of 0.0001. We employ the SGD optimizer. With a batch size of 16, we trained the network independently for each stage over the course of 100 epochs.

4.3 Detection Experiments

Table 2 shows the detection performance of the proposed method and its variants. "5. Segment based" is the main scheme we proposed, which serves as the benchmark in the following discussions. If we do not use the propagation network (bi-LSTM), the detection rate is shown as "1. No Prop." in Table 2. It is essentially a single-frame detector performed on the P/B frames. Therefore, there exist a number of miss detection cases particularly in the nighttime video. The second one, "2. I-Frame only," is the case that the bi-LSTM is used only to fill in the missing I-frame vehicle detection. That is, The bi-LSTM net is applied to the I-frame and its two neighboring frames; it does not apply to the other P/B frames. It recovers some missing cars, and thus, it improves the

Table 2: Variation of components in MFD&T.

Variant	AP_{50}			GFLOPS	Speed (FPS)
	Night	Day	Both		
1.No Prop.	0.653	0.825	0.779	**139**	**80**
2.I-Frame only	0.789	0.874	0.851	148	74
3.ResNet50	0.820	0.882	0.865	166	66
4.Sliding window	**0.823**	**0.883**	**0.867**	186	59
5.Segment based	0.815	0.879	0.862	161	68

detection rate by about 7 percentage points (pp) (in the "Both" case), but the computing speed is lower than "1. No Prop." by 6 FPS (Frames Per Second).

The third variant, "3. ResNet50" is the case in which we use a deeper backbone (ResNet50) for the P/B-Frame detection with segment-based LSTM. The improved detection rate (in the "Both" case) is less than 0.3 percentage points (pp), but the speed is lower by 4 FPS. Instead of segment partition, if we use a sliding window that moves frame by frame, then the sliding window-based LSTM ("4. Sliding window") offers only about 1 percentage point (pp) accuracy improvement over "5. Segment based" but at about an additional 9 FPS speed cost. For the nighttime videos, the segment-based bi-LSTM stage improves the detection rate very significantly, about 16 pp, compared to the "1. No Prop." system. Therefore, we chose the segment-based LSTM with the ResNet18 detector configuration ("5. Segment based") since this combination provides the best trade-off when the speed is a high priority.

Figure 8 shows some successful cases. There are some cases that a car cannot be detected if we use only motion vectors (MV) or only residuals. The purple box shows a car can only be detected if we include the residual information in the detector. The red box indicates the case that a car is detected only when the motion vector is included. In other words, sometimes, one of them contains more information than the other, especially near the border of a fisheye image. The bi-LSTM propagation network is able to recover missing cars based on the detected car in another frame, as shown by a yellow box in Figure 8. The blue boxes show three paused cars that are recovered by the bi-LSTM network. In the previous frames, they ran into a red light, and thus, they paused to wait for the green light. These paused cars cannot be detected in the compressed domain without a propagation network (bi-LSTM) since there is no movement of these cars. By examining many similar successful cases, we have two observations. The tracking ability offered by bi-LSTM net can recover a number of misdetection cases, particularly for the nighttime video. Also, to achieve the best performance, our system should use both MV and residual information. These points are verified by the simulations in the next subsection.

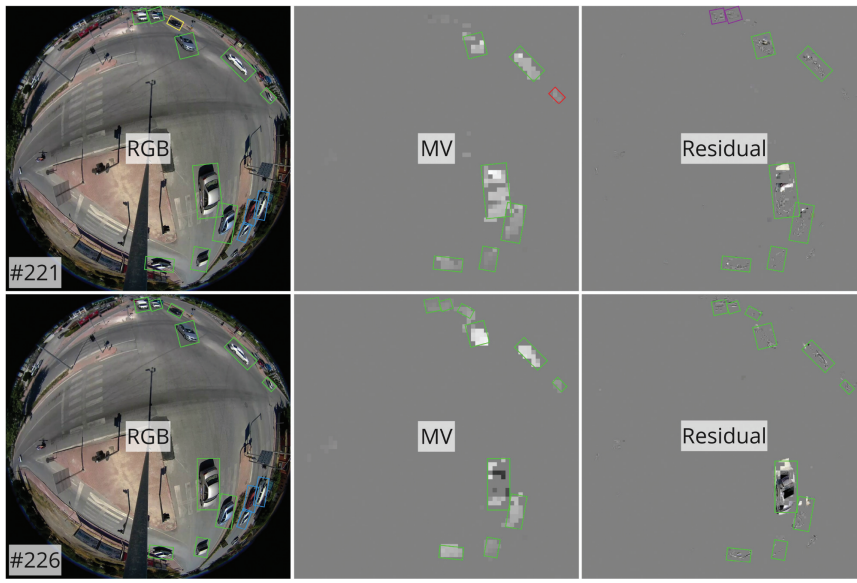


Figure 8: Compressed domain successful examples. Green box: detected cars on P/B-frames. Purple box: cars that are detected only when we also use the residual information. Red box: cars that are detected only when we use motion vectors. Yellow box: the car is not detected at the current frame but recovered from the next frame where it is successfully detected. Blue box: paused car recovered by bi-LSTM.

There are some failure cases that may be further improved, as shown in Figure 9. When the image noise is very high, especially in the nighttime video, the motion field, and residual image are noisy and hard to detect objects (Figure 9(a)). Most cars in this dataset are white cars with white headlights. The headlight reflection also causes difficulty in the compressed domain (Figure 9(b)). The detected bounding box is much larger than the real car. The camera was mounted at a pole about eight meters above the ground. Sometimes a strong wind can shake the camera; it induces a large amount of noise in the compressed data, and the object detector fails (Figure 9(c)).

4.4 Ablation Study

4.4.1 Modality Variation

Table 3 shows the results when we use different modalities for the P/B-frame detector. Using residual images only gives a lower accuracy than the motion vector while their speeds are almost identical. The temporal information is more important than the spatial information in detecting a moving object.

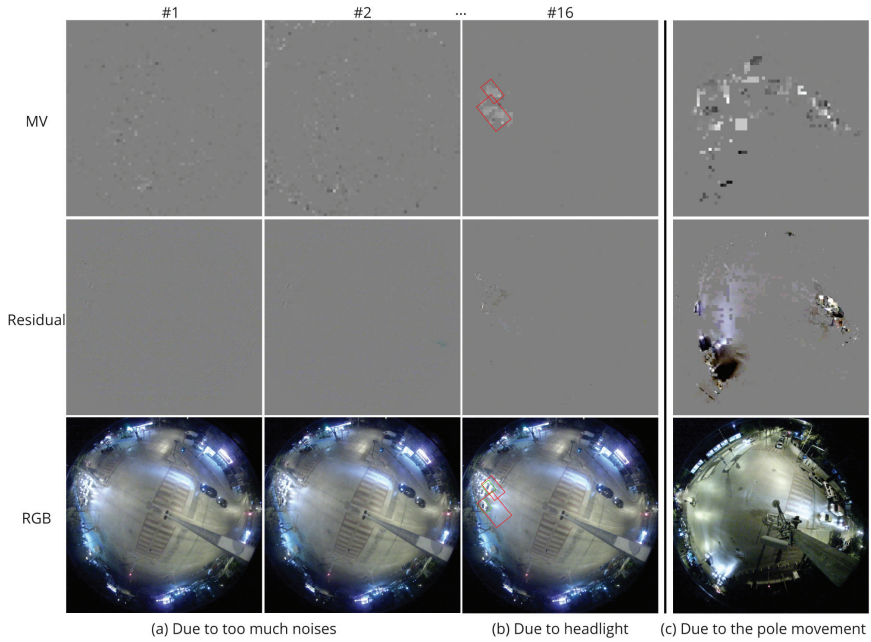


Figure 9: Compressed domain failure examples. (a) Missing detection is caused by too much noise in motion vectors and residuals. Also, the car movement is very small (the car starts moving right after a pause for the traffic light), so the motion information is unclear. (b) The bounding boxes are too big due to the car headlight reflection on the road. Green box: ground truth; Red box: predicted bounding box. (c) Sometimes, a strong wind can shake the pole where the camera is mounted. It induces a large amount of noise in the compressed data.

In this experiment, the bi-LSTM network with (MV+R) input is used in producing Table 3.

When the detector uses both MV and residuals to generate bboxes, we test the propagation network (LSTM) with four variations on its input modalities, and the results are shown in Table 4. In the "Box only" configuration, the inputs to the bi-LSTM network contain only the bbox information. The Box-LSTM-only case lowers the accuracy slightly and improves the detection speed a little bit. Since we use the same backbone for the P/B-frame and for the input port of the Image-LSTM, we stored the image features generated by ResNet18 and fed them directly to the second stage of the Image-LSTM port to save some computations. That is, we skip the ResNet18 in the first stage of the Image-LSTM port. Therefore, the speed is comparable with the box input-only LSTM network. As expected, we chose the MV+R modality for both the R3Det detector and LSTM.

Table 3: Modality variation on the P/B-frame detector in MFD&T.

Modality	**AP$_{50}$**			**GFLOPS**	**Speed (FPS)**
	Night	**Day**	**Both**		
MV only	0.802	0.867	0.850	158	70
R only	0.761	0.843	0.821	157	70
MV+R	0.815	0.879	0.862	161	68

Table 4: Modality variation on the propagation network (LSTM) in MFD&T.

Modality	**AP$_{50}$**			**GFLOPS**	**Speed (FPS)**
	Night	**Day**	**Both**		
Box only	0.810	0.873	0.856	**158**	**70**
MV only	0.814	0.877	0.861	160	69
R only	0.812	0.872	0.856	159	69
MV+R	**0.815**	**0.879**	**0.862**	161	68

4.4.2 GOP Size Variation

We also tested several GOP sizes in H.265 format when encoding our dataset. Table 5 shows the bitrate, PSNR, accuracy, and speed trade-off between these different GOP-size videos. By using a smaller GOP size, the coding bit rate becomes higher (and a slightly higher PSNR in our setting). However, since we only process the P/B-frames, smaller GOP leads to fewer P/B-frames, which leads to a slightly lower accuracy but a slightly faster detection speed. The GOP size of 32 has the highest accuracy but is a bit slower than the GOP size of 8. To achieve a higher compression efficiency, a GOP size of 32 is more often used.

4.4.3 Segment Size Variation

Our propagation network (bi-LSTM) can be expanded to accept different numbers of frames in a segment. The purpose of using bi-LSTM is to track the

Table 5: GOP size vs detection accuracy and speed on MFD&T.

GOP Size	**Bitrate (kB)**	**PSNR (dB)**	**AP$_{50}$**			**Speed (FPS)**
			Night	**Day**	**Both**	
8	1,675	37.90	0.811	0.875	0.858	**70**
16	1,225	37.86	0.812	0.878	0.860	69
32	931	37.68	**0.815**	**0.879**	**0.862**	68

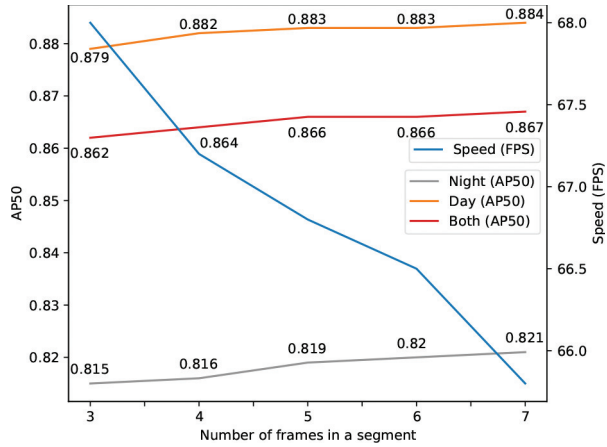


Figure 10: Segment size vs detection accuracy and speed on MFD&T.

cars to reduce false negative and false positive cases. Therefore, a larger segment size generally produces better results but at the expense of computational complexity. Figure 10 shows the trade-off between the number of frames in a segment and the accuracy and speed. Increasing the frame number in a segment slightly enhances the accuracy but lowers the inference speed. In the end, the three-frame setting is chosen as a suitable compromise between accuracy and speed.

4.4.4 Compressed Video Quality

Lower image quality is expected to lower the object detection accuracy. We trained another model at a bit rate of about 10% of the previous coding bitrate, 84 kbps, instead of 931 kbps. The H.265 encoding process performs quite well on this type of video since the camera is not moving, and thus it only needs to encode the moving objects for the most part. Therefore, the PSNR value drops by about 3.3 dB. In this case, the accuracy decreases by about 2 pp to 3 pp. Figure 11 shows the image quality comparison between these two bitrates. It is apparent that the decoded RGB image at a lower bitrate has visible artifacts, but the motion vectors and residual images do not change drastically.

We also conduct experiments using a wide range of compressed bit rates (image quality) as inputs, as seen in Figure 12. The 84K trained model was employed for the low bitrates (<900K), and the 931K model was used for the higher bitrates (>900K). This figure shows an incisive analysis of the performance of our object detection method under varying image quality (bitrates). In Figure 12, the horizontal axis represents bitrates in kbps, and the vertical axis denotes the AP_{50} scores for MF D&T. Three curves are displayed

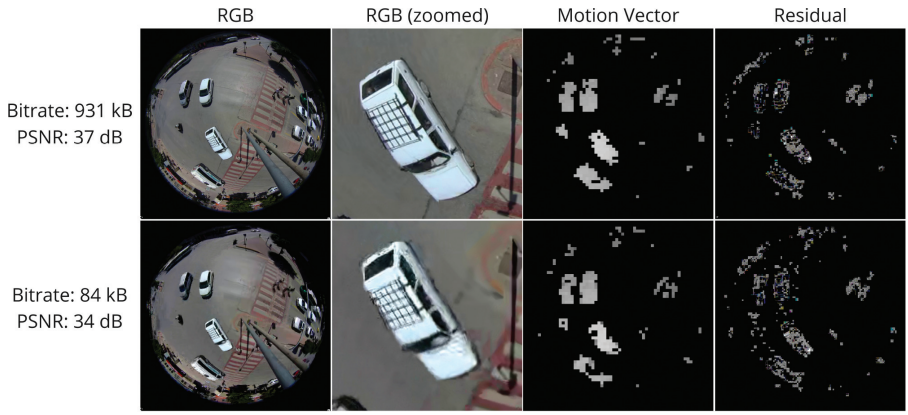


Figure 11: Visual quality comparison at two coding bit rates. At 84 kbps: PSNR = 34.36 dB, AP50 (night, day, both) = (0.782, 0.863, 0.841); at 931 kbps: PSNR = 37.68 dB, AP50 (night, day, both) = (0.815, 0.879, 0.862).

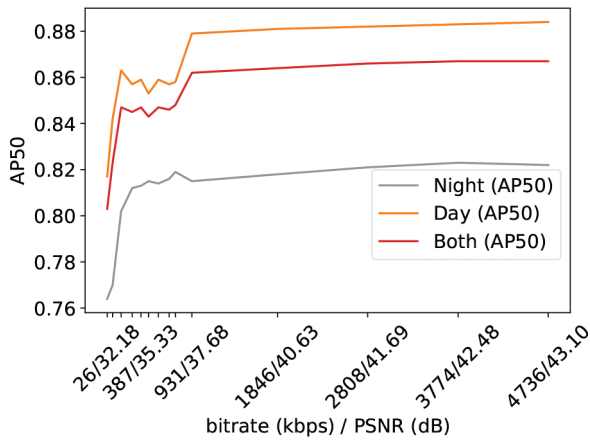


Figure 12: Bitrate (image quality) variations are affecting the detection accuracy of MFD&T.

corresponding to day, night, and combined test data. As the bitrate increases, the AP_{50} scores ascend as well. It is interesting to see a cliff effect around 80 kbps (∼34.3 dB PSNR). The detection accuracy drops dramatically when the video quality is lower than this corner point. On the other hand, the detection accuracy saturates quickly above this point. For all three curves, the detection rate increases only slightly above 800 kbps (∼37.5 dB). Although their implementations are different, this "cliff effect" phenomenon is similar to that of optimal signal detectors in digital communications [36], [25]. When the signal strength surpasses a certain level, the error rate drops dramatically.

Table 6: MFD&T tracking results (MOTA and MOTP [5]).

Method	MOTA	MOTP (IOU)	Speed (FPS)
MFD&T	**86.1**	**85.8**	**68**

4.5 Tracking Experiments

In this method, when the detection task is completed at Stage 2, so is the object tracking. As discussed in Section 4.1, the object detection and the objection tacking tasks are merged in our approach. There is no additional processing needed to generate the tracking result. Table 6 shows the MOTA and MOTP metrics of our method, and they are all above 80 at 68 FPS. In Section 6, we will compare our methods with some popular pixel-domain tracking methods.

5 Scheme 2: Extract Objects from Detected Trails

The method proposed in this section uses the concept of "track to detect". We first construct a "motion trail" image of cars and then detect the car trail. Next, we extract the object bbox in each individual frame. In the pixel domain, frame differences of several frames (a segment) are overlapped to form trail images. This scheme in pixel-domain was proposed in [4]. In the compressed domain, the motion trail is formed by overlapping the motion field/residual image of several frames. The definition of a segment in this section is the same as before. That is, the first and last frames of a segment are KF, and two neighboring segments share a common KF. We extend the pixel-domain track-to-detect (TtoD) scheme in [4] to the compressed domain in this paper.

5.1 Proposed System

The TtoD method has two main stages and an optional third stage, as shown in Figure 13. Stage 1 constructs the motion field/residual trails and then detects the trails using the oriented object detection network, R3Det [39]. Ideally, a detected trail (rotated bbox) contains a single car. In Stage 2, we designed a lightweight multi-head object extractor to generate individual bboxes at different frames for each trail. All bboxes (cars) from the same trail are assigned the same ID. This ID is propagated to the following segments. In the last stage (Stage 3), we use multi-modal bidirectional-LSTM to further improve the detection accuracy.

The trail image is constructed as follows. Similar to the definition of Motion History Image (MHI) [7], the moving pixels of each object in each

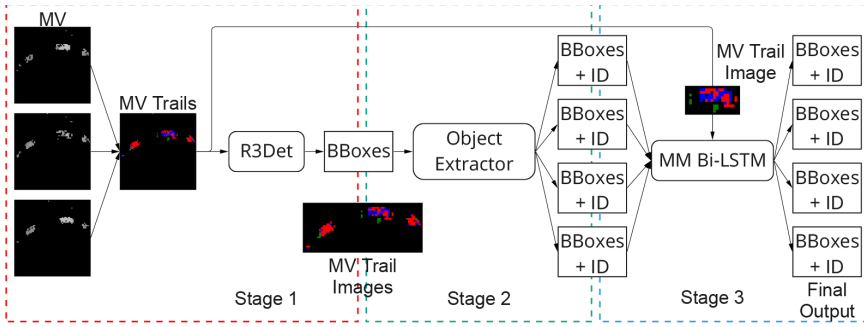


Figure 13: Our overall architecture consists of 3 stages. Stage 1 forms and detects the motion trail. Stage 2 extracts object instances on the individual frame. Stage 3 is optional, which can further improve the AP_{50} accuracy [4].

frame are first identified. This is called *update* function, $D(x, y, t)$. $D(x, y, t)$ is a binary-value function at location (x, y) at frame t. When the motion field or residual image is in use, we first apply the morphological operations (opening and closing) in the image processing software package to reduce the image noise. Then, the non-zero pixels are assigned value 1, which typically indicates the location of moving objects. We thus obtain $D(x, y, t)$ from the given motion field and/or residual image. Then, Equation (3) is used to construct the trail image $H(\cdot)$. The newer object (non-zero) pixels overwrite the previous frames. $H(\cdot)$ is thus assigned quantized grey levels. The motion trails in Figs. 13 and 14 are color-coded (not greyscale value) only for illustration purposes.

$$H(x, y, t) = \begin{cases} t \cdot 255/n & \text{if } D(x, y, t) = 1 \\ H(x, y, t-1) & \text{otherwise} \end{cases} \tag{3}$$

where $H(x, y, t)$ is the greyscale trail for the t-th frame in a segment ($t \in \{1, \ldots, n\}$) and n is the number of frames in a segment. The initial value of $H(x, y, t = 0) = 0$ for all (x, y) pixels.

Our multi-head object extractor network in Stage 2 is shown in Figure 14. The goal of this network is to generate the object bbox in each frame inside a single trail. The input to this multi-head object extractor is the image of one trail detected in Stage 1. The image size is normalized to 64x64. Because the trail image content is rather simple, we use ResNet18 [15] for the backbone, Feature Pyramid Network (FPN) [21] for the neck, and Feature Refinement Module (FRM) from R3Det [39]. The same car ID is assigned to all detected boxes from the same trail. Because two neighboring segments share a common KF, this ID is reliably passed on to the next segment.

The third stage, multi-modal bidirectional LSTM, is the same network used in Section 4. The bbox information and ID generated at Stage 2 are the

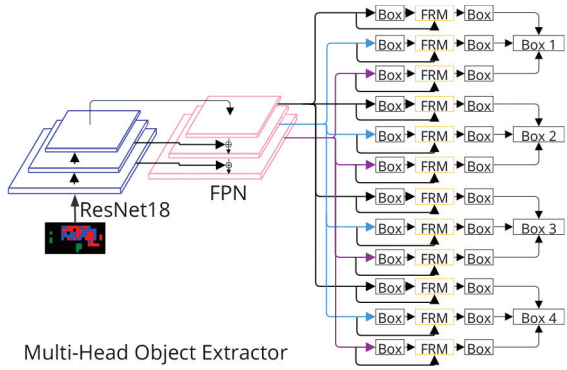


Figure 14: Our multi-head object extractor (Stage 2) [4].

inputs to the Box-LSTM port. The trail image extracted in Stage 2 becomes the input to the Image-LSTM port. Both the upper and lower image ports use the same trail image. The purpose of the third stage is to refine the results (detected bboxes) obtained from Stage 2. Often, Stage 2 already provides very high accuracy. Therefore, Stage 3 can be dropped to reduce computation.

Because the basic components used in this scheme are the same as the ones used in the last MFD&T scheme, its implementation and training procedure are thus similar to that of MFD&T. We train three stages of the TtoD scheme one by one.

5.2 Detection Experiments

In this section, we continue using the ICIP20 VIP Cup dataset for experiments. We use the same settings in the previous section. One segment contains three frames, and the default GOP size is 32 unless stated otherwise.

5.2.1 R3Det IoU Variation

In Table 7, various variants of R3Det using different IoU definitions are compared in terms of mean Average Precision (mAP) and AP on two datasets: DOTA [38], and ICIP [43] (in pixel domain and compressed domain). In this table, the mAP values on the DOTA dataset are cited from Yang *et al.* [42]; the AP_{50} values on the ICIP dataset are generated by our pixel-domain TtoD algorithm (which will be explained in Section 6) and generated by our compressed-domain TtoD algorithm described earlier. That is, in the training phase, the IoU calculation in the first R3Det motion trail detection stage is using different definitions.

Table 7: Detection comparison in R3Det IoU variations.

IOU Method	**mAP DOTA [38]**	**AP_{50} ICIP [43] Pixel Domain**	**AP_{50} ICIP [43] Compressed Domain**
Original R3Det [39]	0.765	0.755	0.762
GWD+R3Det [40]	0.802	0.803	0.798
KLD+R3Det [41]	0.806	0.825	0.828
KFIoU [42]	0.809	0.866	0.871

This table reveals that the original R3Det has been enhanced by incorporating Generalized Wasserstein Distance (GWD+R3Det) and Kullback–Leibler divergence (KLD+R3Det), with both variants showing improvements in mAP/AP across all datasets. Among the methods evaluated, the KFIoU variant achieved the highest performance, suggesting that this method may be particularly effective for object detection tasks.

5.2.2 GOP Size Variation

Table 8 shows the detection accuracy and speed when we change the GOP size of the compressed video inputs. The best accuracy is achieved when we use the GOP size 32 since more frames can be processed (more information extracted); however, it has a slightly lower speed than the smaller GOP size. Smaller GOP size increases the compressed bit rate when the image quality is kept the same. That is, a smaller GOP lowers the compression efficiency. Since the image quality is similar for three GOP sizes, their detection accuracies are about the same. We will investigate the impact of image quality in Subsection B.5.2.5.

Table 8: GOP size vs detection accuracy and speed on TtoD.

GOP Size	**Bitrate (kB)**	**PSNR (dB)**	**AP_{50}**			**Speed (FPS)**
			Night	**Day**	**Both**	
8	1,675	37.90	0.835	0.882	0.869	**56**
16	1,225	37.86	0.836	0.885	0.871	55
32	931	37.68	**0.838**	**0.886**	**0.873**	53

5.2.3 Stage and Modality Variation

Table 9 shows the modalities affecting the detection accuracy on each stage. Motion vector information is more important than residual. However, the detection accuracy increases when we include both of them. Using both

modalities increases the computational complexity slightly (about 2.6%) as compared to using a single modality. We also compare the 2-stage configuration vs. the 3-stage configuration. Stage 3 offers less than a 1 pp improvement in accuracy, but it costs additional 20 GFLOPs or 5 FPS. For the remaining experiments in this section, the default configuration is the 3-stage system. However, in the next section for the pixel-domain vs compressed-domain comparison, the 2-stage configuration is chosen for its superior speed.

Table 9: Stage and modality variation on TtoD.

Stages	**Modality**	**Night**	**Day**	**Both**	**GFLOPS**	**Speed (FPS)**
2	MV Trail	0.823	0.87	0.857	185	60
2	Residual Trail	0.765	0.864	0.837	185	60
2	MV+Residual Trail	0.831	0.883	0.871	190	58
3	MV Trail	0.835	0.876	0.866	198	56
3	Residual Trail	0.784	0.866	0.845	198	56
3	MV+Residual Trail	**0.838**	**0.886**	**0.873**	210	53

5.2.4 Segment Size Variation

In this experiment, we like to find the best number of frames for each modality. Figure 16 shows the detection accuracy and speed trade-off between various compressed domain modalities (motion vector only, residual only, and motion vector + residual). The test system is a 3-stage configuration. It shows that using four frames instead of three frames gives slightly better detection accuracy when using either motion vector or residual alone. When both input modalities are in use, the 3-frame segment has about the same performance as the 4-frame segment. However, the four-frame segment lowers the speed by about 2 FPS in the inference process. In the nighttime video, we observe that there is quite a bit of a drop in detection accuracy when using five frames or more.

5.2.5 Compressed Video Quality

Similar to the MF D&T scheme (Section 4.4.4), the compressed image quality (bitrate) has an impact on the detection accuracy of the TtoD scheme. We explore this issue using the 3-stage system and a wide range of compressed bitrates, as seen in Figure 15. Although the proposed algorithm is different, performance curves similar to Figure 12 are shown here. A cliff effect with a corner bitrate around 80 kbps ($\sim$ 34.3 dB) also appears, and the AP_{50} value reaches a plateau at around 800 kbps ($\sim$ 37.5 dB). This finding is particularly beneficial for resource-limited settings, allowing for efficacious implementations

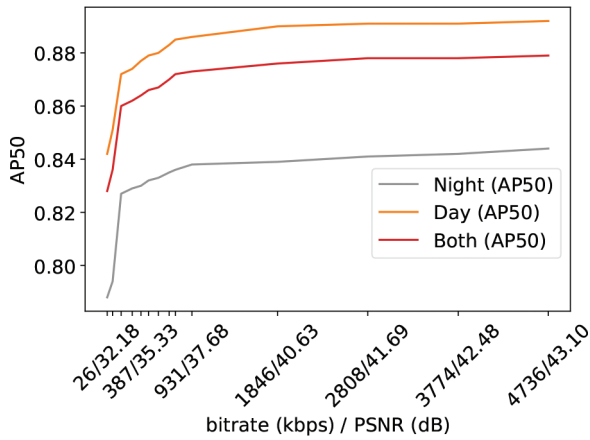
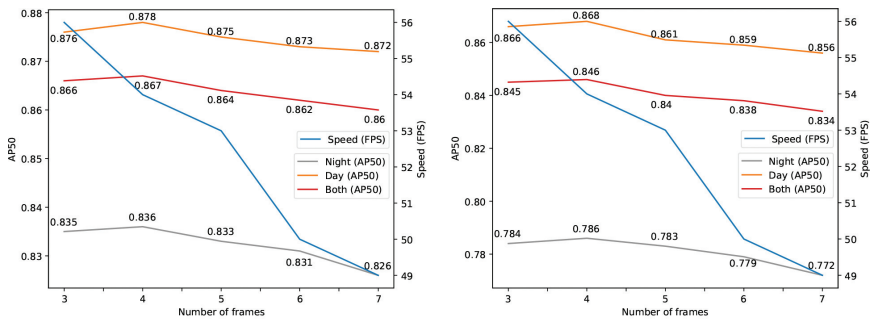
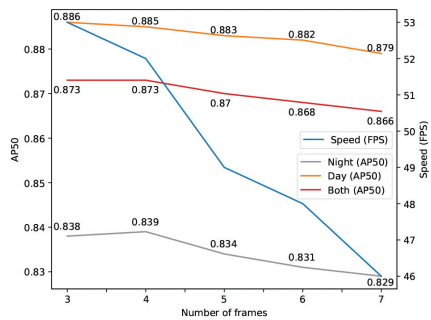


Figure 15: Bitrate (image quality) variations are affecting the detection accuracy of our second scheme, TtoD.



(a) Motion Vector

(b) Residual



(c) Motion Vector + Residual

Figure 16: Modality and segment size vs accuracy and speed on TtoD.

without sacrificing precision. Moreover, Figures 12 and 15 indicate that an image quality of approximately 38 dB is needed for optimal performance in both schemes.

5.2.6 Visualization Examples

Figure 17 shows examples of our track-to-detect approach in the compressed domain. The colors assigned to the MV and residual trails are simply for illustration purposes. In processing, they are quantized greyscale images. Our approach can detect and track the vehicles even in the nighttime video when

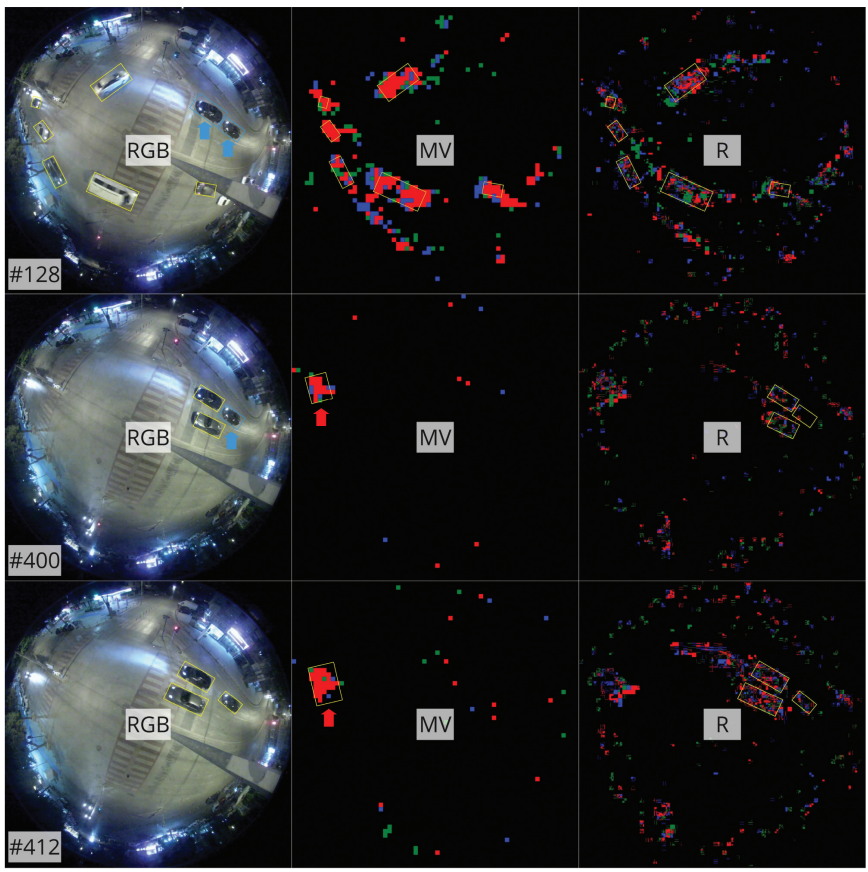


Figure 17: Visual examples of our track-to-detect approach in the compressed domain. (left) RGB, (middle) MV trail, and (right) residual trail modality. Blue arrow: propagated bounding box from other frames, Red arrow: wrong car size due to light reflection noise.

there are more noises compared to the daytime. There are two cars (blue arrows on the left image) waiting for the traffic light in frame #128, and they start moving in frames #400 and #412. Therefore, these cars have no motion information in frame #128. In frames #400 and #412, these cars are moving slowly, and thus their motion vectors are almost invisible. However, these cars can be detected in the residual trails. Typically, the residual trails contain more noise than the MV trails, especially in the nighttime video. But the residual trails can provide complementary information that may not be available in the MV trails. Hence, using both MV trails and residual trails is better than using MV trails alone.

5.3 Tracking Experiments

The tracking results of this track-to-detect approach can be found in Table 10. The 3-stage configuration with a fusion bi-LSTM stage is somewhat better in tracking. Both MOTA and MOTP measurements are above 80. As said earlier, when the detection task is done, the tracking task is completed at the same time. Therefore, it has a very fast speed, 58 FPS. We will compare this method with some high-performance schemes in the pixel domain in the next section.

Table 10: TtoD tracking results (MOTA and MOTP [5]).

Method	**MOTA**	**MOTP**	**Speed (FPS)**
2-stage TtoD	85.8	86.7	**58**
3-stage TtoD	**86.3**	**87.1**	53

6 Compressed-domain vs. Pixel-domain Performance Comparison

We have described two proposed schemes (MFD&T, Compressed TtoD) in the compressed domain in the previous two sections. We like to compare our proposed compressed-domain methods with several popular detection and tracking methods in the pixel domain.

Object detection methods are classified into single-frame image object detection methods and multi-frame video object detection methods, as described in Section 2. For image object detectors, we selected a popular high-performance detector, YOLOv5 [16] as a representative of the regular bbox detector. Also, we chose Rotated YOLOv5 [45] and R3Det [39] for the rotated bbox detectors. Further, we included two leading multi-frame video object detectors, MEGA [9]

and TRA [13] working with SELSA [37]. Previously, we proposed three multi-frame detectors in the pixel domain. They are (1) Bounding Box Propagation method (BBoxProp) in [3], (2) pixel-domain Detect-to-Track (Pixel DtoT) in [2], and (3) pixel-domain Track-to-Detect (Pixel TtoD) method in [4]. These three pixel-domain schemes are outlined below.

The main body of BBoxProp uses classical arithmetic and logical operations, and thus it runs very fast. The tracking operation is performed by propagating bboxes from reliable frames to less reliable frames. The first stage of the pixel-domain DtoT method is a single-frame image object detector, R3Det. Then, a multi-frame motion trail is constructed based on the detected bboxes and the individual trail is detected by R3Det in the second stage. In the third stage, an MM bi-LSTM combines the bboxes generated in the first stage and the trail images generated in the second stage to produce the final object bboxes [2]. Finally, the pixel-domain TtoD structure [4] is similar to that of the compressed-domain TtoD method, except that the input signals are frame differences. That is, the motion trail is constructed using frame differences.

Table 11 summarizes the detection experiments of all schemes when the best model (trained with both daytime and nighttime videos) is in use and the input image resolution is 1024. (A similar performance comparison was reported in [4] but using a down-sampled input image resolution of 640. Therefore, the reported AP_{50} values are slightly lower in [4].) In general, multi-frame video object detectors are better than single-frame object detectors. However, their computational complexity and model size are much larger. For single-frame detectors, rotated bbox detectors are generally better than regular bbox detectors. Our proposed three pixel-domain methods can be considered multi-frame approaches, and indeed, their performance is close to the other video detectors, but their computations are about half of that of the other video detectors. Also, their model size is about 1/3. On the other hand, the compressed-domain detectors have a detection rate slightly higher than their pixel-domain counterparts and have lower computational complexity.

The daytime videos have good image quality, and thus all the multi-frame pixel-domain detectors perform rather well and are pretty close to the best performer, MEGA. However, compared to our proposed three schemes, the MEGA speed is about three times slower, and its model size is three times larger. On the other hand, for the very challenging nighttime videos, the accuracy of Detect-to-Track and Track-to-Detect schemes in the pixel domain is slightly better than that of the best video detector, MEGA (0.78 vs. 0.765). In contrast, because the I-frames are not processed, the compressed-domain schemes have faster speed. Particularly, the Multi-Frame Detection-and-Tracking in the compressed domain is a 2-stage (not 3-stage) scheme; it has the fastest speed, 68 FPS, with slightly lower accuracy. The H.265 encoding process eliminates some noises caused by the low-light environment. Thus, when the compression distortion is small or moderate, the nighttime detection

Table 11: Detection comparison.

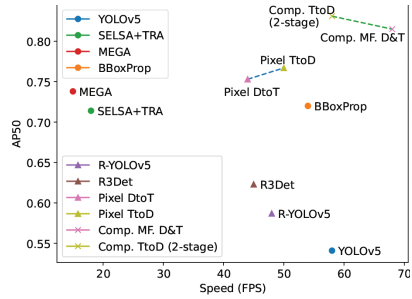
Method	Night AP_{50}	Day AP_{50}	Both AP_{50}	GFLOPS	Speed (FPS)	Model Size (MB)
Pixel domain - Regular bounding box						
YOLOv5 [16]	0.598	0.827	0.772	**220**	**58**	**170**
SELSA [37] + TRA [13]	0.742	0.901	0.842	400	18	520
MEGA [9]	**0.765**	**0.912**	**0.867**	475	15	660
BBoxProp [3]	0.735	0.891	0.857	240	54	180
Pixel domain - Rotated bounding box						
Rotated YOLOv5 [45]	0.587	0.839	0.786	270	48	**160**
R3Det [39]	0.705	0.881	0.812	**251**	45	319
Detect-to-Track [2]	0.782	0.898	0.857	270	44	329
Track-to-Detect [4]	**0.784**	**0.904**	**0.866**	255	**50**	185
Ours - Compressed domain						
Multi-Frame Detect-and-Track	0.815	0.879	0.862	**161**	**68**	205
Track-to-Detect (2-stage)	**0.831**	**0.883**	**0.871**	190	58	**156**

accuracy in the compressed domain is higher than their counterparts in the pixel domain (0.831 vs. 0.784).

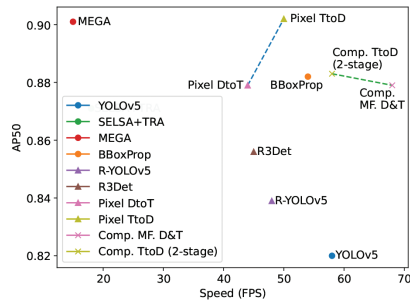
Figure 18 shows our methods compared to the other object detection methods in the accuracy vs speed plot. The methods using the regular bbox are represented by circle markers. The triangle marker indicates the methods with a rotated bbox. The compressed domain approaches are marked by the $\times$ marker. In addition to our proposed two methods in the compressed domain: (a) 2-stage multi-frame detection-and-tracking (Comp. MFD&T), and (b) 2-stage Track-to-Detect in the compressed domain (Comp TtoD), we also include our previously proposed methods in pixel-domain. They are bounding box propagation (BBoxProp) scheme, pixel-domain Detect-to-Track (Pixel DtoT) and Track-to-Detect (Pixel TtoD). Combining the metrics of AP_{50} and speed, our proposed methods are located at the upper-right corner, which implies our schemes have advantages in both accuracy and speed. There is generally a trade-off between speed and accuracy.

Table 12 summarizes a comparison of our tracking methods with the others. The settings (training model and input image size) of our schemes here are the same as those in Table 11. Some post-processing methods can be applied to both regular and rotated bboxes. Again, in addition to the two proposed compressed-domain methods, there are three pixel-domain methods we proposed earlier.

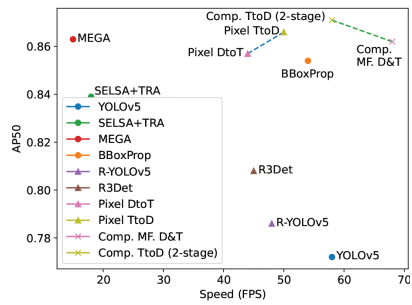
For the pixel-domain conventional schemes, we include several popular methods: Kalman filter [19], SORT [6], DeepSORT [35], FairMOT [47], and ByteTrack [46]. The regular bboxes were generated by YOLOv5 [16], except for FairMOT [47], which is a joint detection and tracking method. The rotated bboxes were generated by R3Det [39]. The BBOXProp method achieves the best accuracy and precision in the category of using regular bboxes, and it is two times faster than the others because we can process several frames at a time



(a) Nighttime video



(b) Daytime video



(c) Both videos

Figure 18: Detection comparison in (a) nighttime video, (b) daytime video, and (c) both videos. A circle marker indicates using a regular bounding box, a triangle marker indicates using a rotated bounding box, and a × marker indicates the compressed domain.

with a lightweight algorithm. The rotated bbox-based methods generally have better performance than their counterparts using regular bboxes. The proposed compressed-domain methods achieve comparable accuracy to the best pixel-domain methods (TtoD and DtoT) but the compressed-domain methods have

Table 12: Tracking comparison.

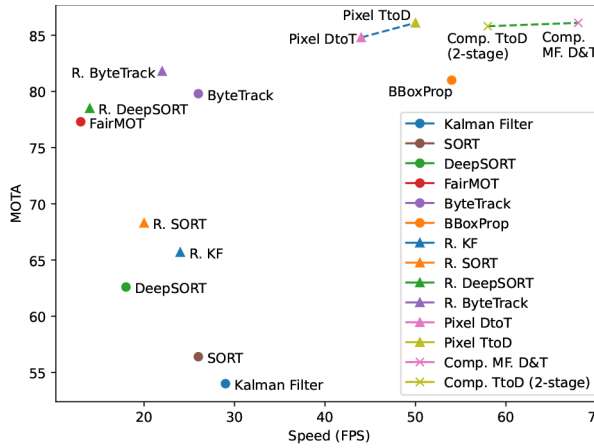
Method	MOTA	MOTP	Speed (FPS)	GFLOPS
Pixel domain - Regular bounding box - YOLOv5 [16]				
KF [19]	54.0	50.1	29	390
SORT [6]	56.4	53.5	26	435
DeepSORT [35]	62.6	60.3	18	627
*FairMOT [47]	77.3	79.2	13	870
ByteTrack [46]	79.8	80.6	26	435
Our BBoxProp [3]	**81.0**	**81.9**	**54**	**209**
Pixel domain - Rotated bounding box - R3Det [39]				
KF [19]	65.7	59.2	**24**	**470**
SORT [6]	68.3	64.6	20	565
DeepSORT [35]	78.5	68.8	14	800
ByteTrack [46]	**81.8**	**82.5**	22	520
Detect-to-Track [2]	84.8	87.1	44	257
Track-to-Detect [4]	**86.1**	**87.5**	**50**	**230**
Ours - Compressed domain				
Multi-Frame Detect-and-Track	**86.1**	85.8	**68**	**170**
Track-to-Detect (2-stage)	85.8	**86.7**	58	190

lower computational complexities. In terms of GFLOPS, compressed-domain MFD&T is lower than the pixel-domain DtoT by 34%.

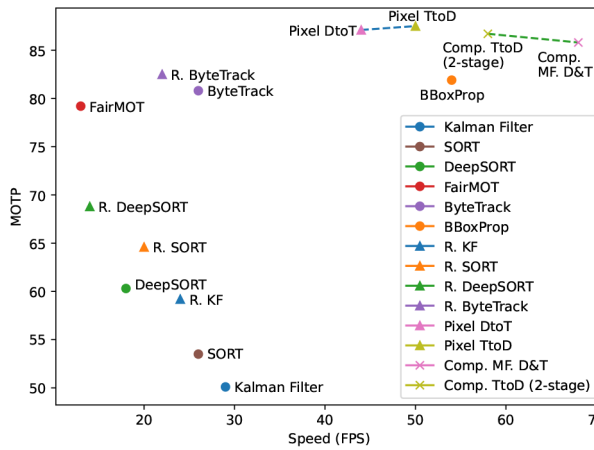
Figure 19 shows various algorithms on the precision vs. speed plots. The labels in this figure are the same as those in Figure 18. Our methods offer up to 4 pp performance improvement as compared to the best ByteTrack and they run about two to three times faster. The enhanced performance comes from the use of a trail that integrates detection and tracking together in our methods. Using a more precise rotated bbox can improve the tracking accuracy of a low-performance tracktor such as the Kalman Filter. However, it has only a marginal effect on high-performance trackers such as ByteTrack [46]. Our method is much faster because we use a lightweight network designed specifically for the fisheye traffic monitoring video.

7 Conclusions

We designed two detection and tracking schemes in the compressed domain using the joint detect-and-track concept to detect and track cars in fisheye monitoring videos. Modern compression standards such as ITU/MPEG H.265



(a) MOTA vs Speed



(b) MOTP vs Speed

Figure 19: Tracking comparison in (a) MOTA and (b) MOTP compared to speed. The circle marker indicates the regular bounding box, the triangle indicates the rotated bounding box and the × marker indicates the compressed domain schemes.

generate compressed bitstreams containing I-frames and P/B-frames. Our schemes process the motion vector and residual information in the P/B-frames. The first method coincides with the "Detect to Track" principle. The conventional DNN-based detector is used to detect single-frame bounding boxes and then the input images together with bounding boxes are fed to a modified multi-modal bi-LSTM network to further track objects and improve

the detection rate. The second method is a design that follows the " Track to Detect" principle. Multiple frame inputs are condensed into a motion trail image, and the trail is detected by a single-frame conventional detector. Then, a multi-class object extractor is designed to extract bounding boxes in the individual frame. The experiment results indicate that our proposed compressed-domain detection and tracking schemes have better performance, particularly on the very challenging nighttime videos than the other DNN-based image and video object detectors and trackers. And our schemes are roughly 1/3 in the model size and 3 times faster in speed as compared to the conventional high-performance video object detectors. In addition, the compressed-domain versions of our schemes can have about equal or better detection and tracking accuracy than their pixel-domain counterparts, and generally, they are faster.

Financial Support

This work was supported in part by the National Science and Technology Council of Taiwan under Grants NSTC-112-2628-E-002-033-MY4, NSTC-109-2223-E-002-005-MY3, and NSTC-112-2634-F-002-002-MBK.

Biographies

Sandy Ardianto (sandyardianto.ee03@nycu.edu.tw) got his doctoral degree from National Yang Ming Chiao Tung University (NYCU), Hsinchu, Taiwan. His research interests include machine learning, computer vision, edge, and distributed computing. He received a BS degree from Sekolah Tinggi Teknik Surabaya in 2013 and worked as Junior Lecturer there for one year. He managed the GPU clusters of the Communication Electronics & Signal Processing Laboratory (CommLab) at NYCU. He worked at IMEC and AnaSystem as a software engineer intern developing deep learning software on edge devices.

Hsueh-Ming Hang (hmhang@nctu.edu.tw) is an Emeritus Professor at National Yang Ming Chiao Tung University (NYCU). From 1984 to 1991, he was with AT&T Bell Laboratories, Holmdel, NJ, and then he was with NYCU from 1991 to 2021. He served as the Dean of the ECE College at NYCU from 2014 to 2017. He was actively involved in the international MPEG standards and his current research interests include deep-learning-based image/video processing and compression. He published over 200 technical papers and held 16 patents. Dr. Hang was an associate editor of the IEEE Transactions on Image Processing and Transactions on Circuits and Systems for Video Technology. He is a recipient of the IEEE Third Millennium Medal and is a Life Fellow of IEEE.

Wen-Huang Cheng (wenhuang@csie.ntu.edu.tw) is a Professor at the Department of Computer Science and Information Engineering, National Taiwan University (NTU). His current research interests include artificial intelligence, multimedia, computer vision, machine learning, digital transformation, and financial technology. He has actively participated in international events and played important leading roles in prestigious journals and conferences, and professional organizations, including Editor-in-Chief for IEEE CTSoc News on Consumer Technology, Senior Editor for IEEE Consumer Electronics Magazine, Associate Editor for IEEE Transactions on Multimedia (T-MM), General Chair for ACM MMAsia (2023), IEEE ICCE-TW (2023, 2022), IEEE ICME (2022) and ACM ICMR (2021), Chair for IEEE Multimedia Systems and Applications (MSA) technical committee, governing board member for IAPR. He has received numerous research and service awards, including the Best Paper Award of 2021 IEEE ICME and the Outstanding Associate Editor Award of IEEE T-MM (2021 and 2020, twice). He is IEEE Distinguished Lecturer, ACM Distinguished Member, and IET Fellow.

References

[1] M. Ahad, A. Rahman, J. Tan, H. Kim, and S. Ishikawa, "Motion history image: its variants and applications," *Machine Vision and Applications*, 23(2), 2012, 255–81.

[2] S. Ardianto, *Efficient Vehicle Detection and Tracking on Fisheye Traffic Monitoring Video in Pixel Domain and Compressed Domain*, Ph.D. Dissertation, National Yang Ming Chiao Tung University, December 2022.

[3] S. Ardianto, H.-M. Hang, and W.-H. Cheng, "Fast Vehicle Detection and Tracking on Fisheye Traffic Monitoring Video Using CNN and Bounding Box Propagation," in *Proceedings of the IEEE International Conference on Image Processing (ICIP), Bordeaux, France*, 2022, 1891–5.

[4] S. Ardianto, H.-M. Hang, and W.-H. Cheng, "Fast Vehicle Detection and Tracking on Fisheye Traffic Monitoring Video using Motion Trail," in *Proceedings of the IEEE International Symposium on Circuits and Systems (ISCAS), Monterey, California, USA*, 2023.

[5] K. Bernardin and R. Stiefelhagen, "Evaluating multiple object tracking performance: the clear mot metrics," *EURASIP Journal on Image and Video Processing*, 2008, 1–10.

[6] A. Bewley, Z. Ge, L. Ott, F. Ramos, and B. Upcroft, "Simple online and realtime tracking," in *Proceedings of the IEEE 2016 International Conference on Image Processing (ICIP), Phoenix, AZ, USA*, 2016, 3464–8.

[7] A. F. Bobick and J. W. Davis, "The recognition of human movement using temporal templates," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 23(3), 2001, 257–67.

[8] Z. Cai and N. Vasconcelos, "Cascade r-cnn: Delving into high quality object detection," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), Salt Lake City, UT, USA*, 2018, 6154–62.

[9] Y. Chen, Y. Cao, H. Hu, and L. Wang, "Memory enhanced global-local aggregation for video object detection," in *Proceedings of the IEEE/CVF Conference Computer Vision Pattern Recognition (CVPR), Seattle, WA, USA*, 2020, 10337–46.

[10] B. Deguerre, C. Chatelain, and G. Gasso, "Fast object detection in compressed JPEG images," in *Proceedings of the IEEE Intelligent Transportation Systems Conference (ITSC), Auckland, New Zealand*, 2019, 333–8.

[11] C. Feichtenhofer, A. Pinz, and A. Zisserman, "Detect to track and track to detect," in *Proceedings of the IEEE International Conference Computer Vision (ICCV), Venice, Italy*, 2017, 3038–46.

[12] F. A. Gers, N. N. Schraudolph, and J. Schmidhuber, "Learning precise timing with LSTM recurrent networks," *Journal of Machine Learning Research*, 3(Aug), 2002, 115–43.

[13] T. Gong, K. Chen, X. Wang, Q. Chu, F. Zhu, D. Lin, N. Yu, and H. Feng, "Temporal ROI align for video object recognition," in *Proceedings of the AAAI Conference Artificial Intelligence (Virtual)*, 2021, 1442–50.

[14] K. He, G. Gkioxari, P. Dollár, and R. Girshick, "Mask R-CNN," in *Proceedings of the IEEE International Conference on Computer Vision (ICCV), Venice, Italy*, 2017, 2961–9.

[15] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR), Las Vegas, NV, USA*, 2016, 770–8.

[16] G. Jocher, A. Stoken, J. Borovec, and et al., "ultralytics/yolov5: v3.1 - Bug Fixes and Performance Improvements," *https://ultralytics.com*, 2020.

[17] R. E. Kalman, "A new approach to linear filtering and prediction problems," *Transactions of the ASME Journal of Basic Engineering*, 82, 1960, 35–45.

[18] H. Kieritz, W. Hubner, and M. Arens, "Joint detection and online multi-object tracking," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops (CVPRW), Salt Lake City, UT, USA*, 2018, 1459–67.

[19] R. Labbe Jr., "Kalman and bayesian filters in python," *GitHub eBook*, 2014.

[20] L. Leal-Taixé, A. Milan, I. Reid, S. Roth, and K. Schindler, "Motchallenge 2015: Towards a benchmark for multi-target tracking," *arXiv preprint arXiv:1504.01942*, 2015.

[21] T.-Y. Lin, P. Dollár, R. Girshick, K. He, B. Hariharan, and S. Belongie, "Feature pyramid networks for object detection," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR), Honolulu, HI, USA*, 2017, 2117–25.

[22] T.-Y. Lin, P. Goyal, R. Girshick, K. He, and P. Dollár, "Focal loss for dense object detection," in *Proceedings of the IEEE International Conference on Computer Vision (ICCV), Venice, Italy*, 2017, 2980–8.

[23] Q. Liu, B. Liu, Y. Wu, W. Li, and N. Yu, "Real-Time Online Multi-Object Tracking in Compressed Domain," *IEEE Access*, 7, 2019, 76489–99, DOI: 10.1109/ACCESS.2019.2921975.

[24] W. Liu, D. Anguelov, D. Erhan, C. Szegedy, S. Reed, C.-Y. Fu, and A. C. Berg, "SSD: Single shot multibox detector," in *Proceedings of the European Conference on Computer Vision (ECCV), Amsterdam, The Netherlands*, 2016, 21–37.

[25] J. G. Proakis, *Digital Communications*, McGraw-Hill, 2000.

[26] J. Redmon, S. Divvala, R. Girshick, and A. Farhadi, "You only look once: Unified, real-time object detection," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR), Las Vegas, NV, USA*, 2016, 779–88.

[27] S. Ren, K. He, R. Girshick, and J. Sun, "Faster R-CNN: Towards real-time object detection with region proposal networks," *Advances in Neural Information Processing Systems*, 28, 2015.

[28] F. Sun, L. Huanyi, L. Zhiyang, X. Li, and Z. Wu, "Arbitrary-angle bounding box based location for object detection in remote sensing image," *European Journal of Remote Sensing*, 54(1), 2021, 102–16, DOI: 10.1080/22797254.2021.1880975.

[29] S. Tomar, "Converting video formats with FFmpeg," *Linux Journal*, (146), 2006, 10.

[30] C. Wang, H. Yang, C. Bartz, and C. Meinel, "Image captioning with deep bidirectional LSTMs," in *Proceedings of the 24th ACM International Conference on Multimedia, Amsterdam, The Netherlands*, 2016, 988–97.

[31] C. Wang, H. Yang, and C. Meinel, "Image captioning with deep bidirectional LSTMs and multi-task learning," *ACM Transactions on Multimedia Computing, Communications, and Applications (TOMM)*, 14(2s), 2018, 1–20.

[32] H. Wang and M. Song, "Ckmeans.1d.dp: optimal k-means clustering in one dimension by dynamic programming," *The R journal*, 3(2), 2011, 29.

[33] S. Wang, H. Lu, and Z. Deng, "Fast object detection in compressed video," in *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV), Seoul, Korea*, 2019, 7104–13.

[34] M. Wien, *High Efficiency Video Coding: Coding Tools and Specification*, Springer, 2015.

[35] N. Wojke, A. Bewley, and D. Paulus, "Simple online and realtime tracking with a deep association metric," in *Proceedings of the IEEE International Conference on Image Processing (ICIP), Beijing, China*, 2017, 3645–9.

[36] J. M. Wozencraft and I. M. Jacobs, *Principles of Communication Engineering*, John Wiley & Sons, 1964.

[37] H. Wu, Y. Chen, N. Wang, and Z. Zhang, "Sequence level semantics aggregation for video object detection," in *Proceedings of the IEEE/CVF International Conference Computer Vision (ICCV), Seoul, Korea*, 2019, 9217–25.

[38] G.-S. Xia, X. Bai, J. Ding, Z. Zhu, S. Belongie, J. Luo, M. Datcu, M. Pelillo, and L. Zhang, "DOTA: A large-scale dataset for object detection in aerial images," in *Proceedings of the IEEE/CVF Conference Computer Vision Pattern Recognition (CVPR), Salt Lake City, UT, USA*, 2018, 3974–83.

[39] X. Yang, J. Yan, Z. Feng, and T. He, "R3det: Refined single-stage detector with feature refinement for rotating object," in *Proceedings of the AAAI Conference on Artificial Intelligence, Vancouver, Canada*, 2021, 3163–71.

[40] X. Yang, J. Yan, Q. Ming, W. Wang, X. Zhang, and Q. Tian, "Rethinking rotated object detection with gaussian wasserstein distance loss," in *Proceedings of the International Conference on Machine Learning (ICML) (Virtual)*, 2021, 11830–41.

[41] X. Yang, X. Yang, J. Yang, Q. Ming, W. Wang, Q. Tian, and J. Yan, "Learning high-precision bounding box for rotated object detection via kullback-leibler divergence," in *Proceedings of the Neural Information Processing Systems (NeurIPS) (Virtual)*, 2021, 18381–94.

[42] X. Yang, Y. Zhou, G. Zhang, J. Yang, W. Wang, J. Yan, X. Zhang, and Q. Tian, "The KFIoU Loss for Rotated Object Detection," *arXiv preprint arXiv:2201.12558*, 2022.

[43] E. Yüncü, M. Alhaddad, B. Karaman, and S. E. Böncü, "VIP cup 2020 at ICIP 2020," *https://signalprocessingsociety.org/community-involvement/vip-cup-2020-icip-2020*, 2020.

[44] M. Zand, A. Etemad, and M. Greenspan, "Oriented Bounding Boxes for Small and Freely Rotated Objects," *IEEE Transactions on Geoscience and Remote Sensing*, 60, 2021, 1–15, DOI: 10.1109/TGRS.2021.3076050.

[45] S. Zhang, X. Wang, P. Li, L. Wang, M. Zhu, H. Zhang, and Z. Zeng, "An Improved YOLO Algorithm for Rotated Object Detection in Remote Sensing Images," in *Proceedings of the IEEE 2021 4th Advanced Information Management, Communicates, Electronic and Automation Control Conference (IMCEC), Chongqing, China*, 2021, 840–5.

[46] Y. Zhang, P. Sun, Y. Jiang, D. Yu, F. Weng, Z. Yuan, P. Luo, W. Liu, and X. Wang, "Bytetrack: Multi-object tracking by associating every detection box," in *Proceedings of the European Conference on Computer Vision (ECCV), Tel Aviv, Israel*, 2022, 1–21.

[47] Y. Zhang, C. Wang, X. Wang, W. Zeng, and W. Liu, "Fairmot: On the fairness of detection and re-identification in multiple object tracking," *International Journal of Computer Vision*, 129(11), 2021, 3069–87.