Interpolation-based Object Detection Using Motion Vectors for Embedded Real-time Tracking Systems

Takayuki Ujiie Masayuki Hiromoto Takashi Sato
Graduate School of Informatics, Kyoto University
Yoshida-hon-machi, Sakyo, Kyoto 606-8501, Japan
paper@easter.kuee.kyoto-u.ac.jp

Abstract

Deep convolutional neural networks (CNNs) have achieved outstanding performance in object detection, a crucial task in computer vision. With the computational intensiveness due to repeated convolutions, they consume large amount of power, making them difficult to apply in power-constrained embedded platforms. In this work, we present MVint, a power-efficient detection and tracking framework. MVint combines motion-vector-based interpolator and CNN-based detector to simultaneously achieve high accuracy and energy efficiency by utilizing motion vectors obtained inexpensively in the environments wherein encoding is conducted at the cameras. Through evaluations using MOT16 benchmark that evaluates multiple object tracking, we show MVint maintains 88% MOTA while reducing detection frequency down to 1/12. An implementation of MVint as a system prototype on Xilinx Zynq UltraScale+ MPSoC ZCU102 confirmed that MVint achieves an ideal 12x FPS compared with a vanilla detection approach.

1. Introduction

Techniques for object detection have drastically improved their performance. They are expected to be used in broad range of real-world applications, such as IoT devices and autonomous driving systems. Among those techniques, convolutional neural networks (CNNs) are intensively studied and have achieved remarkable performance in various object detection tasks. State-of-the-art CNNs employ complex models with a large amount of parameters, which require intensive multiply-accumulate computations. Real-time detection hence has been significantly difficult, but real-time implementations of CNNs are recently realized by the development of efficient models [17, 23, 24] and the application of highly parallelized computational resources, such as GPUs and specialized hardwares [3, 11, 21].

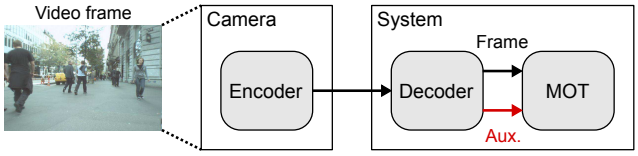


Figure 1. Overview of the target multiple object tracking (MOT) system with a camera that captures encoded videos. The encoded videos are decoded in the MOT system into image frames and auxiliary information such as motion vectors.

With such development of CNNs for object detection, improvement of the performance of multiple object tracking (MOT) [19] has become an active research topic as a more advanced subject. There are offline and online approaches for MOT on videos; as offline trackers, LMP_p [28] and POI [31] are the ones that are specialized for humans as target objects. They not only use CNNs for precisely detecting and tracking human subjects, but the CNNs are also used for pose estimation and temporal object identification between the adjacent frames. As online trackers, SORT [2] and EAMTT [27] utilize CNN or DPM [5] for accurate object detection. By spending large part of the computational costs on the object detection efforts for the spatio-temporal tracking, they realize real-time tracking with moderate tracking performance.

For embedded platforms, however, CNN-based real-time MOT is still a challenging problem because the allowable energy consumption is strictly limited. In this work, we aim to construct an efficient MOT method based on a highly accurate CNN-based detector. To achieve this, we focus on information given by video codecs. In some practical vision systems such as surveillance cameras, video encoders are already equipped with the sensor edges and compressed video stream is transmitted to the servers. In this work, we assume a system that performs MOT for those encoded streams as shown in Fig. 1, which contains a camera with a video encoder and an MOT system. The encoder in the camera is used to reduce the transmission bandwidth to the

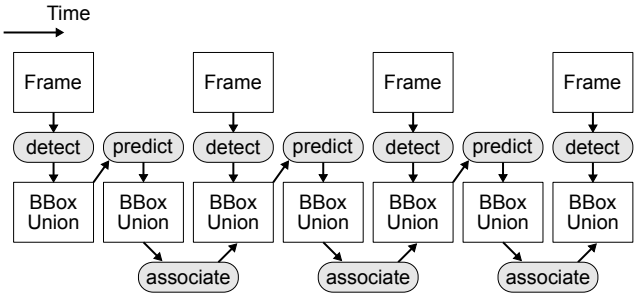


Figure 2. Procedures of the tracking-by-detection. The bounding boxes (BBox Union) are generated by the detection process. Then, a unique ID is annotated to each bounding box. The IDs in BBox Union are propagated along time series by the prediction and association processes.

MOT system by compressing video streams. In this system, we utilize intermediate data obtained from the decoding process as auxiliary information to reduce the computational complexity of the MOT.

There exist researches that utilize intermediate data of video codecs for image recognition. For example, MPEG-flow [15] utilizes motion vectors to efficiently compute the optical flow. MV-CNN [32] combines motion vectors with CNN to realize real-time action recognition. Although the objective of MV-CNN is close to the objective of this paper, our proposed method intends to decrease the computational complexity more aggressively, by interpolating results of object detection.

In this work, we propose a novel method called "MVint" to realize energy-efficient real-time detection by utilizing both CNN and motion vectors extracted from encoded video streams. MVint switches the tracking process according to the type of the frames encoded by video codecs. For I-frames, which are compressed and decompressed independent with other frames, MVint performs accurate detection using CNN. For P-frames, which are compressed and decompressed using the preceding frame, MVint performs low complexity interpolation using the detection results of I-frames, thereby eliminates computationally intensive CNN-based detection while the tracking performance is well maintained. When I-frame exists for every N frames in the video stream, the detection frequency using CNN can be reduced to $1/N$ of the naïve approach in which CNN is used for every frame to realize MOT.

2. Tracking-by-detection

Tracking-by-detection (TBD) [7, 33] is a popular approach for multiple object tracking. In this section, we overview the TBD according to Geiger, *et al.* [7]. The overall process of the TBD is summarized in Fig. 2. The processing pipeline of the TBD is divided into three steps: (1) detection, (2) prediction, and (3) association. The union of the bounding boxes, which are the detection results, are represented as "BBox Union." The bounding boxes are propagated for downstream frames through the prediction and association processes.

First, in the detection process, the bounding boxes of the object proposals are generated from an input frame. To accomplish this, various detectors are used: feature-based detectors such as DPMv5 [26], CNN-based detectors such as R-CNN [10, 9, 25], and single-shot approaches [17, 23, 24]. In the TBD, because the overall tracking performance is determined by the performance of detection, a large part of the computational costs are dedicated to the detection process to achieve good tracking performance.

Next, in the prediction process, the bounding boxes are filtered to remove uncertainty that comes with the transition of the objects. The Kalman filter [29] is the most common method for this purpose. A general form of the systems that Kalman filter targets is denoted as

$$x_{k+1} = Ax_k + Bu_k + w_k \tag{1}$$
$$z_k = Hx_k + v_k \tag{2}$$
$$p(w) \approx N(0, Q) \tag{3}$$
$$p(v) \approx N(0, R). \tag{4}$$

Eq. (1) presents the process model to represent the state transition of the system, Eq. (2) presents the observation model to represent the change made when the state is observed, and Eq. (3) and (4) present probability distributions of the noises, where $N(a, b)$ denotes the normal distribution with mean a and variance b. The variables are the state x_k, the disturbance input (or control input) u_k, the process noise w_k, and the observation noise v_k at time k.

Finally, in the association process, unique IDs are annotated to the bounding boxes after the prediction process, and corresponding boxes are associated to boxes that will be obtained in the next frame. The most common algorithm for the association is the Hungarian method [16], which is known as an algorithm for optimal matching. A cost function $f(\mathrm{BB_A}, \mathrm{BB_B})$, which indicates affinity of the bounding boxes of the adjacent frames, is chosen in advance. For unions $\{\mathrm{BB}_i^X\}_{i=1,\cdots,N}, \{\mathrm{BB}_j^{X+1}\}_{j=1,\cdots,M}$ representing the bounding boxes of the two consecutive frames X and $X+1$, a cost (affinity) matrix is computed. By solving this cost matrix as a complete bipartite graph, the optimal matching to minimize the sum of the cost is obtained. In accordance with the matching results, the IDs of the boxes in frame X are propagated to those in frame $X+1$. The IDs for the boxes that have no matching box are discarded and new boxes in frame $X+1$ are annotated by new IDs.

In section 4.1, we will describe the proposed method by comparing the TBD shown above.

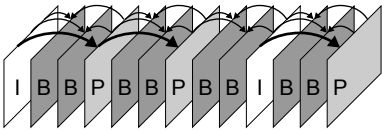


Figure 3. Inter-frame motion compensation. Each frame in the video sequence is encoded by one of the three frame types: I-frame, P-frame, and B-frame.

3. Motion vectors in codecs

We give a brief description of video codecs (especially, MPEG-2 [12]), focusing on the motion vectors that are used by the proposed method. MPEG-2 is one of the video codec standards. In an MPEG-2 encoded video stream, frames are stored as a compressed sequence in a video container. Each frame is divided into small regions (e.g. 16×16) named macroblocks, for which the encoding and decoding processes are conducted. The decoding process includes many sub-processes such as motion compensation, inverse discrete cosine transform (inverse DCT), inverse quantization of DCT coefficients, and variable length decoding.

Among them, motion compensation by inter-frame prediction is a key process in the decoding. The motion compensation is used to improve the efficiency of the compression by exploiting the temporal similarity between the consecutive frames. In the motion compensation, a motion vector, which is a translation offset from the reference frame to the target frame, is utilized to represent the movement of the macroblock. The motion vectors are stored in an encoded stream and extracted when the stream is decoded. Each decoded motion vector is associated with the corresponding macroblock of the reference frame to restore the target frame.

Generally, for the motion compensation defined in video codec standards [12, 14, 13], there are three types of frames having different roles: I-frame, P-frame, and B-frame. The I-frame is treated as a reference frame in motion compensation. It contains complete information to restore the pixel map by itself without using information from other frames. On the other hand, the P-frame refers the previous reference frame to restore the pixel map. The B-frame also refers to other frames, but it uses the subsequent frames as well as the previous frames. The restored frames are also treated as reference frames in motion compensation for other frames. Fig. 3 illustrates an example sequence to show how frames are referred by each other. In the figure, the reference frames and the target frames are connected by arrows in direction of the reference to the target frames. Whereas compression rate increases if a larger number of P-frames and B-frames are contained in a video sequence, while more processing power is required to restore them. Here we define a "group of pictures (GOP)" that contains N frames from a certain I-frame to the next I-frame. We use the GOP as a processing unit in the following sections.

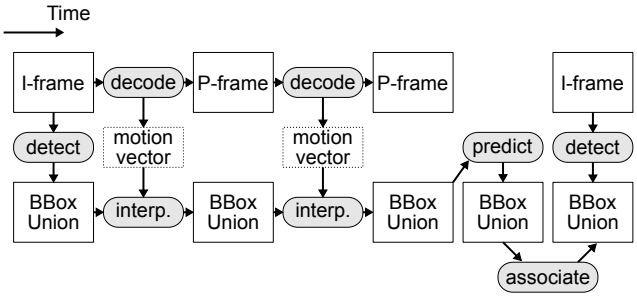


Figure 4. Tracking process of the proposed method (MVint). For I-frames, MVint performs precise detection to generate bounding boxes (BBox Union). For P-frames, MVint performs low-cost interpolation to generate boxes referring boxes of the previous frame. As a next I-frame comes, the interpolated boxes are discarded and new boxes are generated by detection.

4. Proposed method

4.1. MVint framework

Based on the TBD described in Section 2, we propose MVint, which is a computationally efficient detection method for video streams utilizing motion vectors. MVint reduces the frequency of computationally demanding detection process by replacing it with a simple interpolation.

Fig. 4 overviews the tracking process in MVint. The connections of the frames on the upper side of the figure indicate the decoding process by inter-frame prediction. MVint performs different processes depending on the frame types (I or P). Note that in this work we assume the video streams contain only I- and P-frames for simplicity. On I-frames, MVint generates bounding boxes by the detector as done in the conventional TBD approach. It is important to generate bounding boxes thoroughly and accurately at this stage. To that end, MVint may use relatively expensive algorithms in terms of the computational cost, such as CNNs. In this paper, we finally adopt SqueezeDet [30] for our system prototype in Section 6 to realize both efficiency and accuracy.

On P-frames, MVint generates bounding boxes only from the detection results of the previous reference frame without executing detection on the current frame. This process can be regarded as "interpolation" of the location of the bounding boxes from the detection result of the previous frame. To interpolate the bounding boxes, motion vectors extracted from the motion compensation process are utilized. Since MVint targets encoded video streams that only contain I- and P-frames, the interpolation of the bounding boxes is repeated while the P-frames continue. When the next I-frame appears, the interpolated boxes are discarded and new boxes are generated by the detector.

Considering MVint in the context of the TBD, MVint simplifies the detection process by the interpolation using motion vectors. Since there is no constraint for the prediction and the association processes, we can adopt any methods for these processes in the proposed MVint framework. In this paper, we adopt the methods as follows in terms of simplicity and efficiency. First, the prediction process is implemented as identity transformation without addition of any noises; i.e., the prediction does nothing. Then, the association process is implemented by the Hungarian method. As a cost function for the Hungarian method, we utilize

$$f_{\mathrm{IoU}}(\mathrm{BB_A}, \mathrm{BB_B}) = 1 - \frac{|\mathrm{BB_A} \cap \mathrm{BB_B}|}{|\mathrm{BB_A} \cup \mathrm{BB_B}|}, \tag{5}$$

where $|\mathrm{BB}_x|$ represents the area of the bounding box BB_x. Eq. (5) means intersection over union (IoU) of the two bounding boxes $\mathrm{BB_A}$ and $\mathrm{BB_B}$.

4.2. Interpolation

Various implementations for the interpolation of bounding boxes in P-frame can be applied to MVint. To take advantage of the MVint framework, the computational cost for the interpolation should be much smaller than that of the detection process. As a naive implementation, we propose linear interpolation in this paper. Let a set of motion vectors for a bounding box be $\{\boldsymbol{v}(i,j)\}_{i=1,\cdots N, j=1,\cdots M}$, where $\boldsymbol{v}(i,j)$ is a individual motion vector corresponding to each macroblock, and N and M are the numbers of the horizontal and vertical macroblocks (i.e., those of the motion vectors) in the bounding box, respectively. With the linear interpolation, the center of the bounding box $\boldsymbol{c}_t$ is moved to the next center $\boldsymbol{c}_{t+1}$ by a linear function as

$$\boldsymbol{c}_{t+1} = \boldsymbol{c}_t + \alpha \cdot \frac{\sum_i \sum_j \boldsymbol{v}(i,j)}{N \cdot M}, \tag{6}$$

where α is a parameter to control the effect of motion vectors. In this method, we assume that the size of the bounding boxes will not change in the interpolation.

The parameter α can be set considering the filling rate γ_f of the object in the bounding box. As a simple example, let us consider the situation that motion vectors of the object are constant $\boldsymbol{v}$ and $\mathbf{0}$ in other regions. To reflect the movement of the object correctly to the center of the bounding box, α should be set as the reciprocal of γ_f. Although each region has different motion vectors with noises in realistic situations, the filling rate is still convenient to determine α. More simply, α can be set approximately 1 for the objects with tight bounding boxes, and set slightly larger than 1 for the objects with loose bounding boxes.

In order to realize a smooth movement of the bounding boxes, we apply the Kalman filter described in section 2. In Eq. (1)–(4), the box center c_t is assigned to the state x_k and the box movement $c_{t+1} - c_t$ is assigned to u_k. Each matrix in the filter is obviously assigned to $A = B = H = I$, where I is the identity matrix. By following Eq. (1) and (2), when the state transition of the system or the observation of the union of the bounding boxes occurs, the Gaussian noise is supplied to estimate c_t. Here, the state x_k assigned by c_t is reset for every I-frame.

MOTA (↑)	Multiple Object Tracking Accuracy [1].
MOTP (↑)	Multiple Object Tracking Precision.
FAF (↓)	False Alarm per Frame.
MT (↑)	Mostly Tracked.
ML (↓)	Mostly Lost.
FP (↓)	False Positive.
FN (↓)	False Negative.
IDsw (↓)	Identity switch.
FM (↓)	Fragmentations.
Hz (↑)	Processing Frequency.

Table 1. Default indices for MOT Challenge.

5. Tracking performance evaluation

5.1. Dataset

We evaluate MVint with MOT16 [19], which is a benchmark for multiple object tracking. MOT16 provides image sequences and metadata such as a frame rate, a total number of the frames, an image size, and detection results of the default detector DPMv5 [26] for each sequence. Outputs of the trackers are the sizes, positions, and IDs of the bounding boxes for each frame. The dataset is split into training and testing sets. The ground truths are only given for the training split. The indices for the test split are calculated when the results for the test split are submitted on the MOT16 official website [1]. Variations of the proposed method are evaluated on the train split since MOT16 officials recommend to submit only one variation to the website.

To evaluate each method accurately, multiple performance indices are considered in MOT16. The default performance indices are listed in Table 1. The arrows indicate whether the bigger value is desirable (↑) or the smaller is desirable (↓) for each index. The most handy index is MOTA [1], which summarizes the overall tracking performance well. MOTA is calculated as

$$\mathrm{MOTA} = 1 - \frac{\sum_t (\mathrm{FN}_t + \mathrm{FP}_t + \mathrm{IDsw}_t)}{\sum_t \mathrm{GT}_t}. \tag{7}$$

Here, GT is the number of the bounding boxes given as the ground truth, and variable t is the frame number.

[1] https://motchallenge.net/results/MOT16/

5.2. Settings

In the experiment, input sequential images are encoded using FFmpeg, and motion vectors are extracted using FFmpeg libavcodec. For each codec, the GOP-size is fixed to 12 unless otherwise specified. We evaluate two different detectors for comparison: DPMv5 as the default detector mentioned in MOT16, and Faster R-CNN [25] (referred as FRCnn) as an example of a CNN-based detector. For each detector, the detection results are given in advance; the results by DPMv5 are included in the dataset and the results by FRCnn are given by F. Yu, *et al.* [31].

In addition to the proposed method, we evaluate two more methods, which are summarized as follows:

Baseline
Detection: performed for all frames.
Association: performed for all frames.

MVint
Detection: performed only for I-frames. (Interpolation is performed for P-frames.)
Association: performed only at the transitions from P-frame to I-frame.

Worst
Detection: performed only for I-frames. (No interpolation for P-frames.)
Association: performed only at the transitions from P-frame to I-frame.

The baseline and the worst methods are the standard TBD method. In particular, the worst method can be considered as a TBD with a lower detection frequency. The baseline gives the upper-bound performance that MVint may achieve, and the worst method corresponds to the lower bound. In the experiments, MVint is evaluated with two variations of the interpolation methods: the linear interpolation (Linear) and the linear interpolation with Kalman filter (LinearK). The parameter α in Eq. (6) is set to $\alpha = 1.0$ for all the variations of the linear interpolation.

The input video is encoded by the simple profile of MPEG-2. The measurement of the processing frequency targets processes for reading and decoding of frames, tracking and interpolation of bounding boxes; note that the processing time for detection itself is not included as a factor of FPS. Motion vector extraction itself is also not included to FPS and is performed by reading preprocessed data.

5.3. Results

Table 2 shows the evaluation results of the three methods described above. First of all, MVint improves all the indices except for the frequency compared with the worst methods that use DPMv5 and FRCnn. In particular, the IDsw of MVint greatly decreases compared with that of the baseline method. This is because MVint performs the association process only when the target frame is switched from P-frame to I-frame, which results that MVint can obviously track the same object while P-frames are fed continuously. As summarized by MOTA, MVint (LinearK) achieves high MOTA values, which is 91.0% of the baseline with DPMv5 and 88.0% of that with FRCnn.

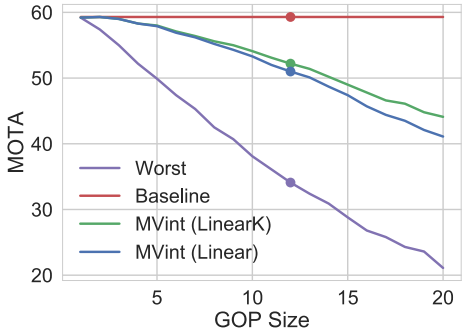


Figure 5. The relationship between the GOP-size and MOTA. The markers are plotted where the GOP-size is 12.

Next, we evaluate the effect of the GOP-size for the tracking performance of MVint. The relationship between the GOP-size and MOTA with the FRCnn detector is shown in Fig. 5. The GOP-size is plotted in the range of 1–20. Note that the GOP-size of 12 is marked in the figure. MOTA decreases almost proportionally as the GOP-size increases. Therefore, the GOP-size can be chosen by considering the trade-off between the capacity of computational resources and the performance of detection and tracking. For example, if the GOP-size is chosen to be 12 as in this evaluation, the CNN detection frequency can be reduced to 1/12 from that of the baseline tracker. Since the computational complexity of the interpolation per bounding box is much smaller than that of the CNN-based detection, the overall complexity will be reduced close to the extent of CNN detection frequency reduction.

We also evaluated the performance of MVint with different variations of video codecs as listed in Table 3. For the evaluation, FRCnn is used for the detection, and LinearK is used for the interpolation. Table 4 shows the evaluation results on the train split. From the results, we confirmed that the tracking performance is affected to some extent by which codec, profile, and level are utilized.

Table 5 summarizes the performance of the proposed method and other methods evaluated on the test split. Other methods are selected from the standard methods [4, 6, 7, 20, 22] mentioned in the MOT16 white paper and from some of the state-of-the-art methods [2, 27, 28, 31] published in the MOT16 official website. Results for the standard methods are listed on the rows under the middle line in Table 5. Input video is encoded by simple profile of MPEG-4 Part 2 and the tracking is performed using FRCnn and Lin-

Method	Detector	MOTA	MOTP	FAF	MT (%)	ML (%)	FP	FN	IDsw	FM	Hz
Baseline	FRCnn	59.3	82.0	1.05	36.9	14.9	5597	37296	2097	1890	55.3
MVint (LinearK)	FRCnn	52.2	78.6	1.92	22.1	23.0	10207	41884	728	1298	38.4
MVint (Linear)	FRCnn	51.0	77.9	2.04	20.9	23.2	10846	42523	729	1419	41.6
Worst	FRCnn	34.1	76.7	3.71	7.2	27.1	19746	51109	1908	2860	58.6
Baseline	DPMv5	27.7	77.3	0.89	7.7	53.6	4707	72606	2487	2586	84.2
MVint (LinearK)	DPMv5	25.2	75.1	1.29	4.4	60.7	6866	75224	512	720	23.4
MVint (Linear)	DPMv5	24.8	74.2	1.33	4.6	60.0	7050	75408	525	787	25.9
Worst	DPMv5	18.2	73.6	1.99	1.2	65.4	10563	78947	811	1350	91.2

Table 2. MOT16 evaluation results (Split: train).

Codec	Library	Profile	Level
H.264	libx264	Baseline	Level 3
MPEG-4 Part2	mpeg4	Simple	Level 1
MPEG-2	mpeg2video	Simple	Main

Table 3. Target codecs.

earK. From the result, we first confirmed that our method with CNN trained by F. Yu, *et al.* [31] has the major improvement of MOTA compared to the standard results by DPMv5. Except for LMP_p, the methods above the middle line are online. To compare MVint with SORTwHPD16 and EAMTT, both of which do not use CNNs for temporal tracking, MVint achieves the second highest MOTA. SORTwHPD16 also uses the detector by CNN trained by F. Yu, *et al.*, and performs detection every frame. With this reason, SORTwHPD16 is considered close to the baseline method in our evaluation. EAMTT uses DT-DPM [5] as the detector, and tracks objects mainly by a particle filter. Due to the accurate CNN detector, our method realized higher MOTA than that of EAMTT by about 2.5 points.

6. System prototype

6.1. Architecture

We consider a tracking system prototype with MVint on an embedded platform to see the performance advantage earned by MVint. This system consists of a multicore processor with a main memory, some specialized coprocessors to compute CNN for detection, an output display, and an input webcam with a video codec encoder.

The entire system architecture is shown in Fig. 6. The white boxes are hardware components and the gray boxes are the processes executed on the hardware. The processes on the multicore processor are implemented to use multiple threads. The input webcam transfers encoded frames to the main memory. The frames are temporarily stored in the main memory and decoded asynchronously in a thread. At

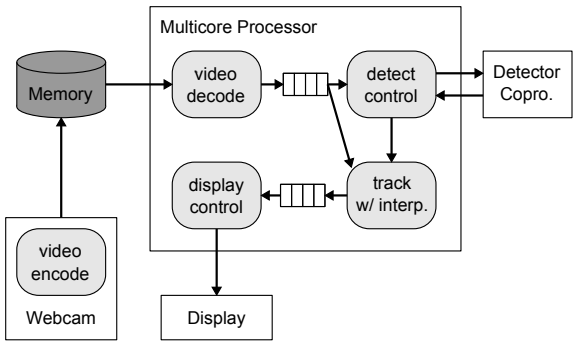


Figure 6. The architecture of the system prototype. Input webcam transfers frames as an encoded stream to the processor. Decoded frames and motion vectors are pushed into a queue to perform detection and tracking asynchronously. The output display receives the tracking results in the appropriate order of sequence.

the same time, motion vectors are also extracted in the decoding process. Certain amounts of frames and motion vectors are pushed into a queue to transfer them to a thread that controls the coprocessor, and to a thread that performs the tracking and interpolation processes. The results are transferred to the output display from the tracking thread in an appropriate order of the sequence.

A more detailed diagram of the frame processing order is shown in Fig. 7. The numbers in each box indicate the frame sequence numbers. The threads have to be synchronized at a time when the processes reach the red lines. As shown in Fig. 7, particularly the display thread has to be synchronized to the track thread to avoid processing earlier than the track thread. On the other hand, the decode thread only has to be synchronized when the detection is completed and can be asynchronously processed in each GOP. At the end of the GOP, all the threads are synchronized to make the next GOP independent. By processing frames as described above, the input frame rate can be gained by GOP-size times faster. When the system's bottleneck is the processing time of the detector as in the case of the CNN-

Codec	MOTA	MOTP	FAF	MT (%)	ML (%)	FP	FN	IDsw	FM	Hz
H.264	51.3	78.4	2.00	19.3	21.9	10636	42368	805	1452	23.4
MPEG-4 Part2	53.5	78.7	1.78	22.6	21.1	9453	41227	648	1222	23.5
MPEG-2	52.2	78.6	1.92	22.1	23.0	10207	41884	728	1298	23.5

Table 4. MOT16 evaluation results for different codecs (Split: train).

Method	MOTA	MOTP	FAF	MT (%)	ML (%)	FP	FN	IDsw	FM	Hz
LMP_p [28]	71.0	80.2	1.3	46.9	21.9	7880	44564	434	587	0.5
POI [31]	66.1	79.5	0.9	34.0	20.8	5061	55914	805	3093	9.9
SORTwHPD16 [2]	59.8	79.6	1.5	25.4	22.7	8698	63245	1423	1835	59.5
MVint (FRCnn LinearK)	55.0	76.7	2.7	20.4	24.5	15766	65297	1024	1594	16.9
EAMTT [27]	52.5	78.8	0.7	19.0	34.9	4407	81223	910	1321	12.2
TBD [7]	33.7	76.5	1.0	7.2	54.2	5804	112587	2418	2252	1.3
CEM [20]	33.2	75.8	1.2	7.8	54.4	6837	114322	642	731	0.3
DP_NMS [22]	32.2	76.4	0.2	5.4	62.1	1123	121579	972	944	212.6
SMOT [4]	29.7	75.2	2.9	4.3	47.7	17426	107552	3108	4483	0.2
JPDA_M [6]	26.2	76.3	0.6	4.1	67.5	3689	130549	365	638	22.2

Table 5. MOT16 evaluation summary (Split: test).

based methods, the detection rate of F FPS can be extended to $G \cdot F$ FPS, where the GOP-size is referred as G. From the evaluation results in Section 5, this method is much better than the case that simply slows down the input frame rate to the maximum frame rate that detector can accept (the "Worst" method), in terms of tracking performance.

6.2. Evaluation

We implement the system prototype described above to evaluate hardware performance in a practical situation. The system is implemented on a commercial system-on-a-chip (SoC), Xilinx Zynq UltraScale XCZU9EG-2FFVB1156, with an evaluation board Xilinx Zynq UltraScale+ MPSoC ZCU102. The software programs are written in C++14 using libavcodec and OpenCV libraries and deployed on Linux running on a quad-core ARM Cortex-A53 processor. We use Logicool HD Pro Webcam C920 as an input webcam, where the video streams are encoded by H.264. The frame rate is 24 FPS and the image size is 320×240. The employed interpolation method is LinearK. Although the original GOP-size of C920 is fixed at 300, we interpret every 12 frames in the GOP as a "sub-GOP" to maintain the tracking performance of MVint. If the GOP-size is virtually shrunk to the sub-GOP-size, the frame at the position of the I-frame may be a P-frame. Therefore, we decode these P-frames and treat them as I-frames in order to virtually realize the GOP-size of 12.

In the coprocessor, we utilized SqueezeDet [30] as a detection network. Although SqueezeDet is relatively a small network compared with other detection networks, we further compressed the network by 8-bit quantization [18] so that it fits in the on-chip memory. Feature maps are computed in a Q8.24 fixed-point number format. Prior to adopting SqueezeDet on the coprocessor, we confirmed that our trained model with these optimizations closely maintain the original detection performance on KITTI benchmark [8].

Process	Time [us]
decode	11714
detect	420495
track	162
display	2574

(a) w/o MVint

Process	Time [us]
decode	304227
detect	423713
track	659
display	6426

(b) w/ MVint

Table 6. A breakdown of the processing time.

We first evaluated the processing time to perform all the processes without MVint. That is, the system performs decoding , detection, tracking, and display. Processes are parallelized similarly as the system with MVint. The evaluation results are shown in Fig. 8a. Fig. 8a represents a part of the frame-processing timeline to show the processing time for some three frames. The interval between the dashed lines indicates the processing time for one frame. The process of the detect thread is conspicuously the bottleneck for the maximum frame rate. To compare the maximum frame rate quantitatively, we show the breakdown of the processing time for a frame between dashed lines in Table 6a. From Table 6a, the maximum frame rate that the system without MVint can process is 1/420495 us=2.38 FPS.

Next, we evaluated the processing time with MVint as described in Section 6.1. The results are shown in Fig. 8b.

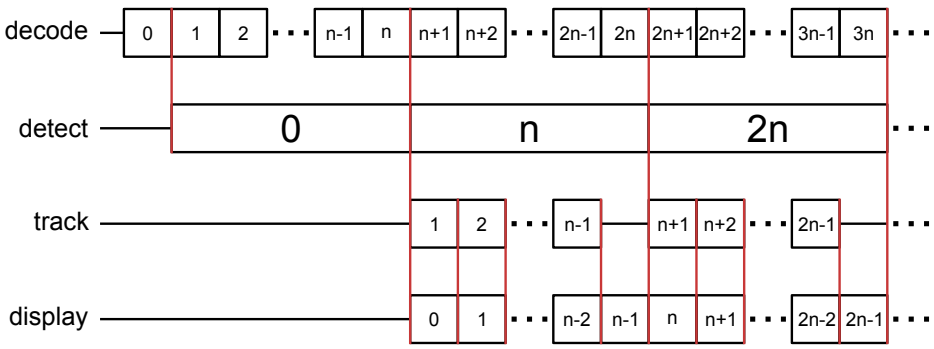
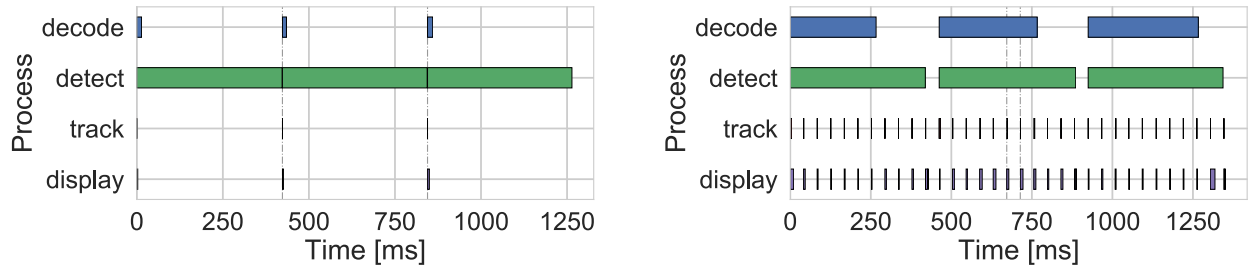


Figure 7. The processing order of frames. Threads have to be synchronized at a time when processes reach red lines. By processing frames in order as here, input frame rate can be gained n times faster.



(a) w/o MVint

(b) w/ MVint

Figure 8. The timelines without and with MVint in tracking. The interval between the dashed lines indicates the processing time to process one frame. Although the processing time for the detection is consistent in both figures, the interval of (b) is much shrunk than that of (a).

The interval between the dashed lines again indicates the processing time for one frame. Although the processing time for detection is consistent with Fig. 8a, the interval for display is shrunk as described in Fig. 7. Table 6b shows the timing breakdown to process one frame. From Table 6b, the maximum frame rate that the system with MVint can achieve is 1/(423713 us/12)=28.32 FPS.

We successfully confirmed that the frame rate of 2.38 FPS has improved to 28.32 FPS by the proposed MVint, which is about 11.90 times acceleration and nearly equal to the ideal value of 12, the sub-GOP-size. We parallelized both the application without MVint and that with MVint to make the most critical detection process be bottlenecks for the maximum frame rate; hence, we directly demonstrated the benefit from MVint in terms of extending the maximum frame rate. Although we fixed the sub-GOP-size to 12 and realized 11.90 times faster frame rate, the maximum frame rate can be adjusted according to the trade-off described in Fig. 5.

7. Conclusion

We proposed MVint, which aims to realize an efficient real-time detection using CNN-based detectors. MVint utilizes motion vectors extracted from video streams encoded by video codec to interpolate bounding boxes by low-complexity operations. Through the evaluations using MOT16, we show that MVint maintains 88% MOTA with a reduction of the detection frequency to 1/12. We further implemented MVint as a system prototype on ZCU102 embedded platform, and confirmed that MVint achieves 28.32 FPS with the parallelized application.

For future works, the exploration of the interpolation methods may be important. Evaluation for the system prototype in this paper implies that there is a room to employ more complex interpolation algorithms. To improve the performance of the interpolator, other intermediate information contained in video streams may be utilized.

Acknowledgment

This work was partially supported by JSPS KAKENHI Grant No. 18H03214.

References

[1] K. Bernardin and R. Stiefelhagen. Evaluating multiple object tracking performance: The CLEAR MOT metrics. *EURASIP*

J. Image and Video Processing, 2008(1), May 2008. 4

[2] A. Bewley, Z. Ge, L. Ott, F. Ramos, and B. Upcroft. Simple online and realtime tracking. In *Proc. IEEE Int'l Conf. Image Processing*, 2016. 1, 5, 7

[3] Y.-H. Chen, T. Krishna, J. Emer, and V. Sze. Eyeriss: An energy-efficient reconfigurable accelerator for deep convolutional neural networks. *IEEE J. Solid-State Circuits*, 52(1):127–138, 2017. 1

[4] C. Dicle, O. I. Camps, and M. Sznaier. The way they move: Tracking multiple targets with similar appearance. In *Proc. IEEE Int'l Conf. Computer Vision*, 2013. 5, 7

[5] P. F. Felzenszwalb, R. B. Girshick, D. McAllester, and D. Ramanan. Object detection with discriminatively trained part-based models. *IEEE Trans. Pattern Anal. Mach. Intell.*, 32(9):1627–1645, 2010. 1, 6

[6] T. E. Fortmann, Y. Bar-Shalom, and M. Scheffe. Multi-target tracking using joint probabilistic data association. In *Proc. the IEEE Conf. on Decision and Control*, pages 807–812, 1980. 5, 7

[7] A. Geiger, M. Lauer, C. Wojek, C. Stiller, and R. Urtasun. 3D traffic scene understanding from movable platforms. *IEEE Trans. Pattern Anal. Mach. Intell.*, 36(5):1012–1025, 2014. 2, 5, 7

[8] A. Geiger, P. Lenz, and R. Urtasun. Are we ready for autonomous driving? The KITTI vision benchmark suite. In *Proc. IEEE Conf. Computer Vision and Pattern Recognition*, pages 3354–3361, 2012. 7

[9] R. Girshick. Fast R-CNN. In *Proc. IEEE Int'l Conf. Computer Vision*, pages 1440–1448, 2015. 2

[10] R. Girshick, J. Donahue, T. Darrell, and J. Malik. Rich feature hierarchies for accurate object detection and semantic segmentation. In *Proc. IEEE Conf. Computer Vision and Pattern Recognition*, pages 580–587, 2014. 2

[11] V. Gokhale, J. Jin, A. Dundar, B. Martini, and E. Culurciello. A 240 G-ops/s mobile coprocessor for deep neural networks. In *Proc. IEEE Conf. Computer Vision and Pattern Recognition Workshop*, pages 682–687, 2014. 1

[12] ISO/IEC 13818-2:2013. Information technology – Generic coding of moving pictures and associated audio information – Part 2: Video. Standard, 10 2013. 3

[13] ISO/IEC 14496-10:2014. Information technology – Coding of audio-visual objects – Part 10: Advanced Video Coding. Standard, 9 2014. 3

[14] ISO/IEC 14496-2:2004. Information technology – Coding of audio-visual objects – Part 2: Visual. Standard, 6 2004. 3

[15] V. Kantorov and I. Laptev. Efficient feature extraction, encoding and classification for action recognition. In *Proc. IEEE Conf. Computer Vision and Pattern Recognition*, 2014. 2

[16] H. W. Kuhn and B. Yaw. The Hungarian method for the assignment problem. *Naval Res. Logist.*, 2(1-2):83–97, 1955. 2

[17] W. Liu, D. Anguelov, D. Erhan, C. Szegedy, S. Reed, C.-Y. Fu, and A. C. Berg. SSD: Single shot multibox detector. In *Proc. the European Conf. on Computer Vision*, pages 21–37, 2016. 1, 2

[18] M. Abadi, et al. TensorFlow: Large-scale machine learning on heterogeneous systems. https://www.tensorflow.org/, 2015. 7

[19] A. Milan, L. Leal-Taixé, I. D. Reid, S. Roth, and K. Schindler. MOT16: A benchmark for multi-object tracking. *Computing Research Repository*, abs/1603.00831, 2016. 1, 4

[20] A. Milan, S. Roth, and K. Schindler. Continuous energy minimization for multitarget tracking. *IEEE Trans. Pattern Anal. Mach. Intell.*, 36(1):58–72, 2014. 5, 7

[21] N. P. Jouppi, C. Young, N. Patil, D. Patterson, et al. In-datacenter performance analysis of a tensor processing unit. In *Proc. ACM/IEEE Ann. Int'l Symp. Computer Architecture*, pages 1–12, 2017. 1

[22] H. Pirsiavash, D. Ramanan, and C. C. Fowlkes. Globally-optimal greedy algorithms for tracking a variable number of objects. In *Proc. IEEE Conf. Computer Vision and Pattern Recognition*, pages 1201–1208, 2011. 5, 7

[23] J. Redmon, S. K. Divvala, R. B. Girshick, and A. Farhadi. You only look once: Unified, real-time object detection. In *Proc. IEEE Conf. Computer Vision and Pattern Recognition*, pages 779–788, 2016. 1, 2

[24] J. Redmon and A. Farhadi. YOLO9000: Better, faster, stronger. *Proc. IEEE Conf. Computer Vision and Pattern Recognition*, pages 6517–6525, 2017. 1, 2

[25] S. Ren, K. He, R. Girshick, and J. Sun. Faster R-CNN: Towards real-time object detection with region proposal networks. In *Proc. the Neural Information Processing Systems*, pages 91–99, 2015. 2, 5

[26] M. A. Sadeghi and D. Forsyth. 30Hz object detection with DPM V5. In *Proc. the European Conf. on Computer Vision*, pages 65–79, 2014. 2, 4

[27] R. Sanchez-Matilla, F. Poiesi, and A. Cavallaro. Online multi-target tracking with strong and weak detections. In *Proc. the European Conf. on Computer Vision*, pages 84–99, 2016. 1, 5, 7

[28] S. Tang, M. Andriluka, B. Andres, and B. Schiele. Multiple people tracking by lifted multicut and person re-identification. In *Proc. IEEE Conf. Computer Vision and Pattern Recognition*, pages 3539–3548, 2017. 1, 5, 7

[29] G. Welch and G. Bishop. An introduction to the Kalman filter. Technical report, Chapel Hill, NC, USA, 1995. 2

[30] B. Wu, F. N. Iandola, P. H. Jin, and K. Keutzer. SqueezeDet: Unified, small, low power fully convolutional neural networks for real-time object detection for autonomous driving. In *Proc. IEEE Conf. Computer Vision and Pattern Recognition Workshop*, pages 129–137, 2017. 3, 7

[31] F. Yu, W. Li, Q. Li, Y. Liu, X. Shi, and J. Yan. POI: Multiple object tracking with high performance detection and appearance feature. In *Proc. the European Conf. on Computer Vision*, pages 36–42, 2016. 1, 5, 6, 7

[32] B. Zhang, L. Wang, Z. Wang, Y. Qiao, and H. Wang. Real-time action recognition with enhanced motion vector CNNs. In *Proc. IEEE Conf. Computer Vision and Pattern Recognition*, pages 2718–2726, 2016. 2

[33] H. Zhang, A. Geiger, and R. Urtasun. Understanding high-level semantics by modeling traffic patterns. In *Proc. IEEE Int'l Conf. Computer Vision*, pages 3056–3063, 2013. 2