Real-Time Detection and Tracking Using Hybrid DNNs and Space-Aware Color Feature: From Algorithm to System

Liang Feng[1(✉)], Hiroaki Igarashi[2], Seiya Shibata[2], Yuki Kobayashi[2], Takashi Takenaka[2], and Wei Zhang[1]

[1] Hong Kong University of Science and Technology, Kowloon, Hong Kong
{lfengad,wei.zhang}@ust.hk

[2] NEC Corporation, Kawasaki, Kanagawa, Japan
h-igarashi@hf.jp.nec.com, s-shibata@ax.jp.nec.com, y-kobayashi@hq.jp.nec.com, takenaka@aj.jp.nec.com

Abstract. Object detection and tracking are vital for video analysis. As the development of Deep Neural Network (DNN), multiple object tracking is recently performed on the detection results from DNN. However, DNN-based detection is computation-intensive. In order to accelerate multiple object detection and tracking for real-time application, we present a framework to import the tracking knowledge into detection to allow a less accurate but faster DNN for detection and recover the accuracy loss. By combining different DNNs with accuracy-speed trade-offs using space-aware color information, our framework achieves significant speedup (6.8×) and maintains high accuracy. Targeting NVIDIA Xavier, we further optimize the implementation from system and platform level.

Keywords: DNN · Object detection · Tracking · GPU

1 Introduction

Multiple object detection and tracking is a key technology for video interpretation. Tracking-by-detection has become the leading paradigm in multiple object tracking due to the recent progress in object detection. Objects are detected each frame as bounding boxes and tracked by matching detections for the same object across frames. Deep neural networks (DNN) for object detection proposed in recent years, such as Faster-R-CNN [18], SSD [15], Yolo [17], Mask-R-CNN [11], etc., provide highly accurate detections, and thus allow simpler but more efficient tracking-by-detection approaches. However, such computation-intensive large DNNs are not fast enough to satisfy the real-time processing, especially when with limited computing power like in embedded systems. Therefore, in addition to these large DNNs, smaller DNN structures are explored for high-speed detection, such as Tiny-Yolo [17], Tiny-SSD [21], etc., although their accuracy is too low to satisfy the detection and tracking requirement.

To simultaneously achieve high speed of the small DNN and high accuracy of the large DNN, we creatively combine both DNNs with different speed-accuracy trade-offs in a time-interleaving way. Relying on the tracking knowledge, the accurate information from large DNN is used to recover the accuracy loss from small DNN. By importing tracking knowledge into detection, the accuracy requirement for detection is relaxed to allow the high-speed small DNN used in most video frames. In this way, both good accuracy and high speed are achieved simultaneously in our framework.

To match detections across frames for tracking, their similarity needs to be measured. Intersection over union (IOU), feature description neural network (NN), etc. are used in recent works [4–6,20,24] for similarity measurement. However, they are either not accurate enough or with high computation complexity. We propose a space-aware color feature for more accurate similarity measurement by extracting both color and space information with high accuracy and low computation complexity. Such a distinguishable feature also allows re-identifying the same object after occlusion and works well in recovering the accuracy of small DNN detections. By combining many novel techniques in detection matching and accuracy recovering, our framework achieves state-of-the-art detection and tracking accuracy at high speed. The framework can also be used in detection-only case to speed up detection while maintaining high accuracy.

Our key contribution is combining hybrid DNNs with different speed-accuracy trade-offs in a time-interleaving way and importing the tracking knowledge into detection, which result in both high accuracy and high speed. The novel usage of the space-aware color feature is another main contribution. In addition, many new techniques are designed to fit these novel concepts. Besides algorithm level, we also perform optimization at system and platform level for a higher-speed implementation, by exploring architectural heterogeneity, multi-core parallelism, data precision, clock frequency, etc., targeting the underlying NVIDIA Xavier. In summary, we design a complete detection and tracking framework with both high accuracy and high speed, from the algorithm level to the system and platform level, from software to hardware. We achieve a high speed at 55.2 FPS for the whole real-time multiple object detection and tracking task, which is 6.8$\times$ faster with similar level high accuracy than the traditional large DNN only method on NVIDIA Xavier.

2 Related Work

2.1 Multiple Object Tracking

MOT can be formulated as a global optimization problem that processes entire video batches at once, which cannot be used in real-time applications, such as flow network formulations-based [25] and probabilistic graphical models-based MOT works [23]. Multiple Hypothesis Tracking (MHT) [13] and Joint Probabilistic Data Association (JPDA) filters [9] used in traditional MOT are still impractical for real-time processing due to the large delay from high combinatorial complexity. Some tracking works build appearance models [3,22] through online learning, which are quite complex. Relying on accurate DNN detection, recent works adopt simple IOU to match detections from frames using Hungarian or greedy algorithm [4–6]. Although with high speed, these works show low accuracy in many scenarios due to little object feature extraction. Further, some works use additional DNNs to describe object feature for matching [20,24], where the feature description NN brings high computation complexity and is hard to train for different scenarios. We propose a simple but efficient space-aware color feature with new matching algorithms to achieve simpler usage, higher speed and even better accuracy. Moreover, current DNN detection-based MOT works only consider one large detection DNN with low speed, which cannot satisfy real-time requirement especially when with limited computing power as in embedded applications. We creatively combine hybrid DNNs for detection to achieve both superior speed and accuracy.

2.2 Color Feature

Color feature distinguishes object efficiently. Many works describe object for tracking using color histogram [1,16]. More complex color features have been proposed, like color naming [19], color attributes [7]. Color feature has also been combined with other well engineered features like gradient [8], HOG [26], correlation filters [7], etc. Due to the power of DNN detection in our work, color histogram is a good choice with simple computation to distinguish objects. Because color feature is most discriminative for objects within the same class from DNN, while other features like edges, etc., have already been included by DNN. Different from existing color features, our novel space-aware color feature uses partial histograms to include space information for better discrimination.

3 Whole Framework

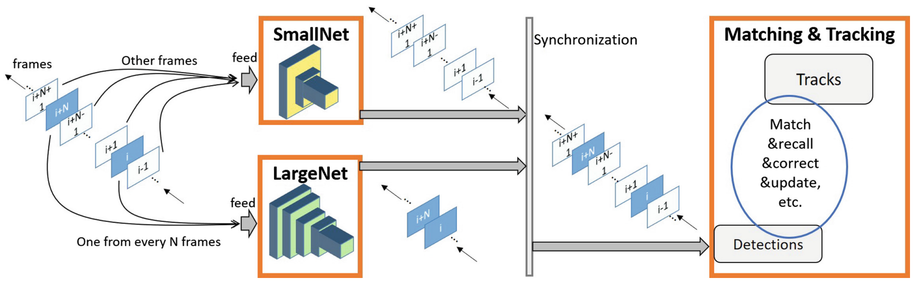

Fig. 1. Whole framework

The framework is based on NVIDIA DeepStream on Jetson Xavier platform. As in Fig. 1, there are three stages, LargeNet, SmallNet and Matching&Tracking (M&T). Each stage is implemented as a pipeline stage in DeepStream and handled by different threads for high parallelism and full resource usage. The stages

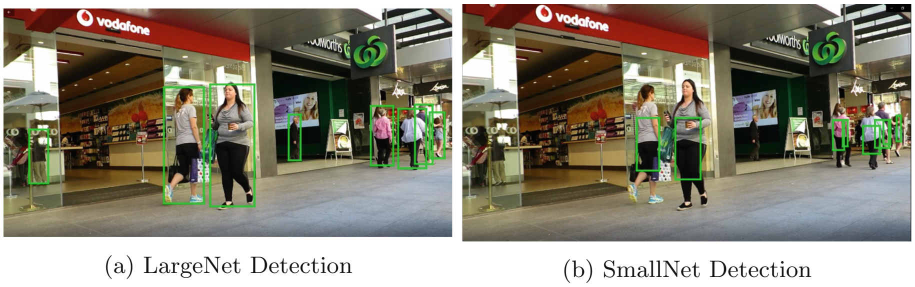

Fig. 2. Detection example for ADL-Rundle-6 from 2D MOT15 benchmark

execute different video frames simultaneously in a pipeline fashion. A synchronization mechanism ensures the frames from SmallNet and LargeNet enter M&T in order. LargeNet uses a large DNN for object detection with low speed and high accuracy, while SmallNet adopts a small DNN for detection with high speed and low accuracy. Among every N frames, only the first frame goes to LargeNet for detection, while the following N-1 frames all go to SmallNet. N is the network switching interval. M&T receives the detection results for each frame in order and performs the tracking. It matches current detections with existing tracks. SmallNet and LargeNet execute on GPU while M&T executes on CPU to fully utilize the heterogeneous architecture.

4 Detection

While LargeNet gives accurate detections, detections from SmallNet are usually with bad bounding boxes and imprecise positions. We use Yolo as the LargeNet and Tiny-Yolo as the SmallNet. Other detection neural networks such as SSD, Faster-RCNN, Mask-RCNN, etc. can also be used as LargeNet. Besides changing the network structure, network compression techniques such as channel pruning [10], quantization [12], etc. can also be used to derive a SmallNet. As shown in Fig. 2, LargeNet detects perfect bounding boxes for most objects. While the bounding boxes from SmallNet usually only cover part of the object with imprecise center point, and many of them are redundant. Some objects cannot even be detected by SmallNet. However, SmallNet can be 6$\times$ faster than LargeNet when running in DeepStream. By using LargeNet every N-th frame while using SmallNet for remaining frames, high speed can be achieved in our framework. The detections from SmallNet will be corrected using tracking knowledge with previous LargeNet detections to recover the accuracy. Due to the low quality of SmallNet, a high detection confidence threshold will lose detection of many objects causing a high false negative number. Therefore in practice, the confidence threshold for a valid detection in SmallNet should be set low to provide more candidate SmallNet detections although imprecisely. More candidates mean more opportunities to find the exact matching of the same object to existing tracks. And the imprecision can be corrected using tracking knowledge with previous LargeNet detections.

5 Matching and Tracking

5.1 Similarity Sorting-Based Matching

The aim of tracking is to match the detections for current frame with existing tracks. Unlike previous IOU based matching [4–6], we rely on both space-aware color feature and IOU for matching. We denote the detections and tracks to be matched as D and T, respectively. A similarity distance s_{ij} is derived for every pair of $t_i \in T$ and $d_j \in D$ based on the space-aware color feature and IOU, where a smaller distance indicates a larger similarity between the detection and track. The smaller the similarity distance is, the more likely the detection should be matched to the track. Therefore, we propose a sorting-based matching to match the most similar pairs with priority as in Algorithm 1. All s_{ij} are sorted in ascending order. From the smallest s_{ij}, we match d_j to t_i if both have not been matched before. Besides, matching is performed only if the similarity distance is small enough ($< Th_{sim}$). $Assign_i$ reflects which detection is matched to track t_i. $Assign_i$ as -1 means no matched detection, while other value j means matching detection d_j to t_i.

Algorithm 1. Similarity Sorting-based Matching

```
   Input: $D$ of size $m$, $T$ of size $n$
   Output: $Assign$ of size $n$
1  Function $SortMatching(D, T)$:
2      all $Assign \longleftarrow$ -1;
3      foreach $t_i \in T$ do
4          foreach $d_j \in D$ do
5              $s_{ij} \longleftarrow SimilarityDistance(t_i, d_j)$;
6          end foreach
7      end foreach
8      sort $\{s_{ij} | 0 \leq i < n, 0 \leq j < m\}$ in ascending order;
9      foreach $s_{ij}$ in ascending order do
10         if $Assgin_i = -1$ && $\forall Assign \neq j$ && $s_{ij} < Th_{sim}$ then
11             $Assign_i \longleftarrow j$;
12         end if
13     end foreach
14     return $Assign$.
15 End Function
```

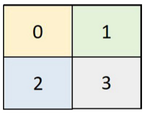

Fig. 3. Partial Histogram Calculation

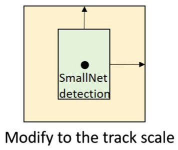

Fig. 4. Scaling for SmallNet Detection

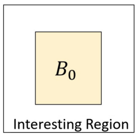

Fig. 5. Interesting Region

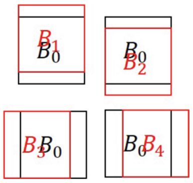

Fig. 6. Bounding Box Derivation

Space-Aware Color-Based Similarity Distance. The similarity distance s_{ij} between track t_i and detection d_j is defined as Eq. 1. If the last bounding box of t_i and the bounding box of d_j have no overlap (IOU = 0) or the detection class of d_j from LargeNet differs from the track class of t_i, s_{ij} is set to a large value MAX to disable matching since t_i and d_j are unlikely to be the same object. Otherwise, the color histograms of t_i and d_j will be used for calculating s_{ij} to check their similarity using space-aware color feature. Each detection or track has 5 color histograms $h_0 \sim h_3$ and H. H is the color histogram of the whole bounding box, while $h_0 \sim h_3$ are the color histograms of four partial boxes $0 \sim 3$ as in Fig. 3, respectively. Lab color space is used for calculating the color histograms by considering all three channels. For a track and a detection, correlation distances between them for H and $h_0 \sim h_3$ are calculated, respectively, before summed up with different weights α_0 (0.1) and α_1 (0.6). Correlation distance is 1 minus the correlation $Corr$ between two histograms. In this way, both color and space information are considered as a space-aware color feature to estimate the similarity with much better accuracy. All 5 histograms of a track t_i are updated in the same way every time a LargeNet detection d_j is matched to it as in Eq. 2, where part of history information is kept for stability. Since SmallNet detection is inaccurate, it will not update the histograms of a track. The class of a track is set as the class of its associated LargeNet detections because the class prediction from LargeNet is usually true. We still check the color histograms for a SmallNet detection even if its class differs from the track, since they are still possible to be the same object due to the frequent wrong classification of SmallNet.

$$s_{ij} = \begin{cases} MAX \text{ , if } IOU(t_i, d_j) = 0 \mid\mid LargeNet\ class\ differs \\ \sum_{k=0}^{3} \alpha_0(1 - Corr(h_{k_{t_i}}, h_{k_{d_j}})) + \alpha_1(1 - Corr(H_{t_i}, H_{d_j})), \\ \qquad \text{otherwise} \end{cases} \tag{1}$$

$$H_{t_i} = (1 - \beta)H_{t_i} + \beta H_{d_j} \tag{2}$$

$$(width\ or\ height)_{t_i} = (1 - \gamma)(width\ or\ height)_{t_i} + \gamma(width\ or\ height)_{d_j} \tag{3}$$

SmallNet Detection Scaling. SmallNet detections usually hold wrong bounding box scale (width and height), and thus we perform scaling to them. Each track holds a scale, which is only updated when a LargeNet detection is matched to it since only scale from LargeNet detection is accurate. The scale is updated as Eq. 3 to consider the history information for stability. The scale can also be updated using a Kalman filter although which performs not stably in real applications. When calculating the similarity distance between a SmallNet detection and a track, only the center point of the detection is kept while the scale of the detection is replaced by the scale of the track to form a new detection bounding box as in Fig. 4. The calculation for color histograms of the detection and IOU will follow the new bounding box. Every SmallNet detection will perform different scaling for calculating the similarity distance with different tracks. In this way, the scale inaccuracy of SmallNet detections can be corrected by the accurate scale from LargeNet detections.

5.2 Space-Aware Color-Based Bounding Box Refinement

Although the center point (position) of SmallNet detection is usually close to the ground truth, but not highly precise. We use space-aware color feature to refine the position of the SmallNet detections which have been matched to a track. The bounding box of the SmallNet detection first performs scaling as in Sec. SmallNet Detection Scaling to change its scale to the track scale. This bounding box is the root bounding box B_0. We consider an Interesting Region centering at center of B_0 with larger scale ($K\times$ width and height, K can be 2) as in Fig. 5. Inside this region, we are aimed at finding a new bounding box position with larger similarity to the matched track, which should be more precise than B_0. The ground truth bounding box is usually near B_0, thus only the Interesting Region needs to be considered. B_0 is moved one small step to four directions, up, down, left and right, respectively, to derive four new bounding boxes $B_1 \sim B_4$, as in Fig. 6. B_0 is parent and $B_1 \sim B_4$ are child bounding boxes. One step for up or down directions and for left or right directions are calculated as $height_{B_0}/M$ and $width_{B_0}/M$, respectively. M is the granularity for refinement and using higher M is more accurate with more computation. M can be set to 8. From each newly derived bounding box, four child bounding boxes of it are further derived. In this way, a graph can be formulated as in Fig. 8. A bounding box stops deriving its child bounding boxes if it is outside the Interesting Region or has appeared

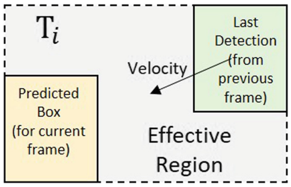
T_i
Last Detection (from previous frame)
Velocity
Predicted Box (for current frame)
Effective Region

Fig. 7. Effective region

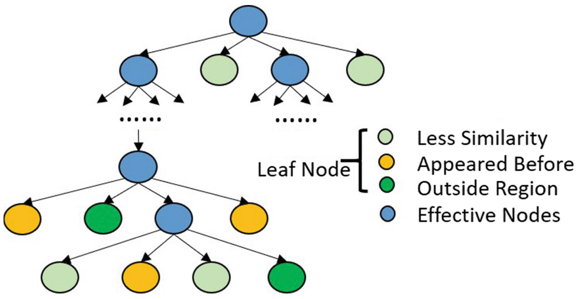

Fig. 8. Bounding box graph

before, so it becomes the leaf node in the graph. If the similarity distance to the matched track of a bounding box increases compared to the distance to the track of its parent bounding box, it also becomes the leaf node; since it indicates a bad direction with less similarity and thus any exploration along this direction is no need and pruned. In this way, the space of the bounding boxes to be searched is reduced for saving computation. Either breath-first-search (BFS) or depth-first-search (DFS) is performed to traverse the graph. During traverse, the similarity distance to the track of each bounding box is calculated as in Sec. Space-aware Color-based Similarity Distance. After traverse, the bounding box with the smallest similarity distance will be chosen as the refined bounding box since it matches the track best. The bounding box of both the SmallNet detection and the track at this frame will be updated as this refined bounding box. The refinement is only performed to a SmallNet detection when its detection confidence is less than Th_{ref} (0.8). The bounding box position is usually highly precise for a SmallNet detection with very high confidence, thus we only refine the detections with low confidence for saving computation. The refinement is only performed for a completed matching while the scaling is needed for calculating the similarity distance as the base of matching.

5.3 Retire-Recall Mechanism

Targeting the occlusion case, we propose a retire-recall mechanism. A newly instantiated track is active and it will become retired if not matched with any detection for Th_{ret} frames. A retired track is permanently deleted if not matched with any detection for Th_{del} ($Th_{del} > Th_{ret}$) frames. Only active tracks are valid and output for the detection and tracking task, while a retired track can be recalled to become active if a detection without matching in active tracks has large similarity to it.

Trajectory Prediction. For each track, the trajectory is recorded as {center point.x, center point.y, frame number} for all its matched detections. Linear regression is applied to the last L trajectory records to calculate the slope as the velocity, in x and y directions, respectively. The center pointer of the track in one frame can be predicted based on its last detected center point and the velocity, and its bounding box can be predicted by combining the predicted center point and the scale of the track. We found linear regression performs more stably than Kalman filter or decay model for trajectory prediction in practice.

Similarity Distance for Retired Tracks. To recall a retired track to be active, the similarity distance between each retired track and each detection which cannot match to any active track is calculated. Then the sorting-based matching algorithm is applied as in Algorithm 1. Calculation of similarity distance for retired tracks is basically the same as Eq. 1, except that the IOU between a retired track t_i and a detection d_j will be the IOU between the Effective Region of t_i and the scaled bounding box of d_j. As in Fig. 7, the Effective Region of t_i is the rectangle covering the bounding box of t_i from its last matched detection and the linear regression-predicted bounding box for t_i at current frame. The object has high possibility to re-appear between its last appeared position and its predicted current position, thus only the detections in the Effective Region should be focused on. In this way, the number of detections requiring color histogram and correlation calculation is reduced for saving computation.

5.4 Whole Matching and Tracking Algorithm

Combining all above techniques, the whole matching and tracking algorithm for M&T stage is as in Algorithm 2. First, the similarity sorting-based matching guided by the space-aware color-based similarity distance as in Algorithm 1 is performed to the detections from current frame and the existing active tracks from last frame, for a largely improved tracking accuracy. Second, for remaining unmatched detections and active tracks, IOU-based Hungarian matching similar to previous works [4,5] is applied to find additional matchings with large overlap, where a matching is valid only if its IOU is large enough ($> Th_{iou}$). In this way, our new space-aware color guided matching is combined with IOU-based matching to take advantages of both. Third, the space-aware color guided similarity sorting-based matching as in Algorithm 1 is applied again to the remaining unmatched detections and existing retired tracks to recall some retired tracks to become active for the retire-recall mechanism as in Sect. 5.3 and match them with corresponding detections. Next, a new active track is created for each remaining unmatched detection from LargeNet, while a new active track is created for a remaining SmallNet detection only if its detection confidence is larger than a high threshold Th_{crt} (0.8). Because only a SmallNet detection with high confidence indicates an object accurately. Then, according to the number of frames each track has not been matched with any detection for, some active tracks retire and some retired tracks are deleted as in Sect. 5.3 for the retire-recall mechanism. The bounding box at current frame of an matched active track will be updated as the bounding box of its matched detection, after scaling and refinement if necessary. Then, for an unmatched active track, traditional tracker such as KCF is applied to decide its bounding box at current frame as a supplement. If KCF fails, its current bounding box will be updated using the trajectory prediction as in Sec. Trajectory Prediction as a further supplement. The updated bounding boxes at current frame of all active tracks with their track IDs will finally output from the whole framework. Algorithm 2 is performed for each frame. We eliminate the re-calculation of the same color histogram and correlation if calculated before by keeping them to reduce the computation. To better utilize the multi-core CPUs, OpenMP is used in many steps along the execution for multi-threading, including similarity distance calculation, bounding box scaling and refinement, trajectory prediction, KCF, etc. The calculation for different tracks or detections are mostly independent and thus suitable for multi-threading with high parallelism.

Algorithm 2. Whole Matching and Tracking Algorithm at Each Frame

```
Input: current frame detections $D$, existing active tracks $T_a$, existing retired
       tracks $T_r$
Output: updated active tracks $T_a$, updated retired tracks $T_r$
1 Similarity sorting-based matching to $D$ & $T_a$;
2 IOU-based Hungarian matching to remaining unmatched $D$ & $T_a$;
3 Similarity sorting-based matching to remaining unmatched $D$ & $T_r$ to recall
  some retired tracks to become active tracks ($T_r \rightarrow T_a$);
4 Initiate new active tracks in $T_a$ for remain unmatched $D$ from SmallNet with
  confidence $> Th_{crt}$ or from LargeNet;
5 Delete tracks from $T_r$ and move tracks from $T_a \rightarrow T_r$ according to the number
  of frames a track has not been matched with any detection for;
6 Update $T_a$ using its matched $D$ after necessary scaling and refinement;
7 Update $T_a$ without matching using KCF or trajectory prediction.
8 return $T_a, T_r$ for current frame.
```

Table 1. Overall MOT metrics

	MOTA↑	**FPS↑**	IDsw↓	FM↓	FP↓	FN↓	MOTP↑
Large-IOU	**47.0**	**8.1**	1733	1178	7663	11717	73.9
Large-Ours	**50.5**	**8.1**	367	1168	7661	11715	74.0
Small-Ours	**−17.3**	**46.3**	85	407	10347	36401	75.8
Hybrid-Ours	**44.5**	**37.2**	513	1308	4626	17024	71.1

6 Experiment Evaluation

The framework is implemented on NVIDIA Xavier platform based on DeepStream, TensorRT, OpenCV, etc. We test the performance on diverse videos in a real-time execution scenario, where 11 videos are generated from the training set of 2D MOT15 benchmark [14]. Yolo-v3 pre-trained on COCO dataset is used as LargeNet and Tiny-Yolo-v3 pre-trained on COCO dataset is used as SmallNet. The neural network switching interval N is set to 10. All experiments are performed on Xavier for a fair comparison.

6.1 Overall Performance

We use standard MOT metrics, whose details are in [2], to evaluate the performance of the detection and tracking task as in Table 1. ↑ in the table means the larger is the better while ↓ on the contrary. MOTA is the overall score to evaluate the detection and tracking accuracy, while FPS reflects the speed. The other MOT metrics are not important for the whole detection and tracking task compared to MOTA and most of them have been covered in MOTA. Large-IOU adopts LargeNet at each frame with IOU-based tracking, similar to previous works [4–6]. Large-Ours applies LargeNet at each frame with all our proposed techniques except bounding box scaling and refinement. Small-Ours uses SmallNet at each frame with our techniques. Hybrid-Ours is our proposed complete framework using hybrid neural networks in a time-multiplexing way with all our proposed techniques. All methods are implemented on Xavier for a fair comparison. Only using SmallNet gives an extremely bad accuracy but with a high speed. While only using LargeNet achieves good MOTA accuracy but with low speed, which cannot satisfy real-time requirement. Our framework with hybrid networks achieves both good MOTA and high speed. It increases the speed by 4.6× to LargeNet-only case, but maintains the similar level MOTA. By importing LargeNet information, it corrects the totally messy results of SmallNet-only case but still maintains a high speed. Furthermore, Large-Ours increases MOTA by 3.5 than Large-IOU and reduces ID switches (IDsw) by 78.8%, which shows our techniques, including space-aware color-based similarity, sorting-based matching, retire-recall mechanism, etc., achieve better tracking ability. Because the space-aware color feature used is more distinguishable and the retire-recall works well for occlusion case. Although targeting on different videos, the MOTA we achieve is at the similar level with other state-of-the-art works and even larger than the best 2D MOT 2015 results on the MOT Challenge website. Our method achieves state-of-the-art accuracy and high speed satisfying real-time requirement.

6.2 Technique Effect

Table 2. Effect of techniques

	MOTA↑	**FPS↑**	IDsw↓	FM↓	FP↓	FN↓	MOTP↑
No-partial-color	**38.5**	**38.2**	514	1733	6074	17942	70.1
No-refinement	**40.5**	**38.3**	600	1547	5369	17721	70.4
No-recall	**44.0**	**37.3**	660	1308	4627	17024	71.1
All(Hybrid-Ours)	**44.5**	**37.2**	513	1308	4626	17024	71.1
Merge	**44.5**	**17.0**	513	1308	4626	17024	71.1
No-OMP	**44.5**	**31.2**	513	1308	4626	17024	71.1

Algorithm Level. To see the effect of different algorithm level techniques we propose, three variations are derived from the complete framework. No-partial-color only uses one whole color histogram H to calculate the similarity distance. No-refinement does not apply the bounding box refinement as in Sect. 5.2. No-recall does not recall any retired tracks. All, the same as Hybrid-Ours, means our complete framework with all proposed techniques. As in Table 2, MOTA is decreased by 6, 4 and 0.5 without partial color histograms, bounding box refinement and retire-recall, respectively. Our novel usage of partial color histograms brings the largest benefit since both the space and color information can be considered into similarity estimation. Bounding box refinement also brings large benefit by correcting the SmallNet detection position. Retire-recall mechanism shows the largest effect in reducing ID switches, by 22.3% since it can recall retired tracks with an existing ID. All these techniques benefit the accuracy but with only little speed overhead and maintain the same level FPS.

System Level. At the system level, we design a highly pipelined structure as in Fig. 1 for the whole framework. If we merge the LargeNet, SmallNet and M&T into the same pipeline stage, the average speed will degrade to 17.0 FPS as Merge in Table 2. Our highly pipelined design brings 2.2× speedup by exploring the parallelism and fully utilizing all computing resources. In addition, we use OpenMP for multi-threading in the CPU execution as in Sect. 5.4. After disabling OpenMP, the average speed degrades to 31.2 FPS as No-OMP in Table 2. Using OpenMP brings 19.2% speedup by fully utilizing multi-core CPUs.

6.3 Platform-Specific Exploration

Table 3. FPS regarding platform-specific exploration

GPU-FP32 (Default)	GPU-INT8	GPU-FP16	DLA-FP16	Best-version	Default+HF-mode	Best-version+HF-mode
37.2	48.7	46.0	45.8	48.9	40.0	55.2

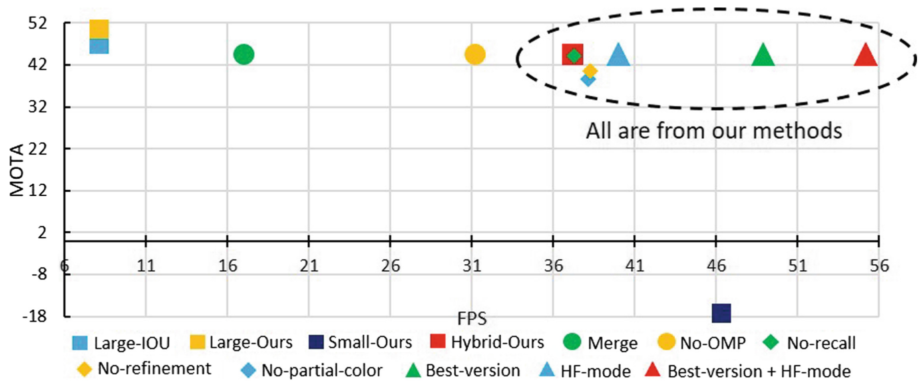

Fig. 9. Accuracy (MOTA) vs Speed (FPS)

Implementation Alternatives. TensorRT on Xavier platform provides different data precisions, including FP32, FP16 and INT8. Besides GPU, the neural network can also be implemented on the Deep Learning Accelerator (DLA) although only some layers supported. For both LargeNet and SmallNet, we try four implementations, GPU-FP32, GPU-FP16, GPU-INT8 and DLA-FP16. The default implementation in previous sections is GPU-FP32. The TensorRT we use only supports FP16 for DLA and the DLA-unsupported layers are executed on GPU. Calibration is used in GPU-INT8 to minimize the information loss. MOTA maintains at the same level as the implementation changes. Because

FP16 is enough to keep most information of the neural network and the calibration for INT8 is effective to minimize the information loss. Using low precisions, the speed is improved due to simpler computation. The highest achievable speed when varying the network implementations for each video respectively is 48.9 FPS on average, denoted as Best-version case, increased by 31.5% compared to the default GPU-FP32 case. The LargeNet is the speed bottleneck since varying the SmallNet implementation does not change the speed obviously while varying the LargeNet does. The average speed of whole framework is 37.2, 46.0, 48.7 and 45.8 FPS, when LargeNet adopts GPU-FP32, GPU-FP16, GPU-INT8 and DLA-FP16, respectively, as in Table 3. Using INT8 gives the largest speedup and two FP16 cases also show large speedup. GPU-FP16 is slightly better than DLA-FP16, since GPU executes faster than DLA and additional data transfer between layers executed on DLA and GPU occurs in DLA-FP16. When only using LargeNet at each frame, the average speed is 15.2, 15.4 and 16.8 FPS for DLA-FP16, GPU-FP16 and GPU-INT8, respectively. When only using SmallNet at each frame, the average speed is 47.9, 52.3, 59.3 FPS for DLA-FP16, GPU-FP16 and GPU-INT8, respectively. Our LargeNet-SmallNet hybrid framework achieves close speed to SmallNet-only case even for DLA-FP16, GPU-FP16 and GPU-INT8. The highest average speed for LargeNet-only case is 16.8 FPS from GPU-INT8, which is still outperformed by our hybrid framework by 2.9×.

High Frequency Mode. Xavier platform provides high frequency (HF) mode with higher clock frequency for computing engines. At this mode, our framework with default case, where both LargeNet and SmallNet adopt GPU-FP32, can achieve 40.0 FPS with a 2.8 FPS improvement as in Table 3. The Best-version case FPS can achieve 55.2 at HF mode with a 6.3 FPS improvement. Considering HF mode and different implementations of the networks, our framework can achieve 55.2 FPS while maintaining the MOTA of about 44.5, which is 6.8× faster than the LargeNet-only baseline with similar level high MOTA. Such a high FPS is beyond the real-time requirement and outperforms most existing works. From the MOT Challenge website, the speed we achieve ranks 12% among all state-of-the-art 2D MOT 2015 results and only 3 out of 89 state-of-the-art MOT 2016 results are faster than us. And we achieve much better accuracy than the works faster than us. Although targeting on different videos from these results, we can conclude that our framework achieves both state-of-the-art speed and accuracy.

6.4 Discussion

The detection and tracking accuracy in terms of MOTA and the speed in terms of FPS for most above evaluated cases are plotted in Fig. 9. All points in the top right are from our methods, which means our proposed methods achieve both high speed and accuracy, especially compared to the LargeNet-only and SmallNet-only cases as Large-IOU, Small-Ours, etc. Even though disabling some techniques, our framework still achieves a large improvement compared to the

SmallNet-only and LargeNet-only cases considering both accuracy and speed, as in No-recall, No-partial-color and No-refinement. Combining all proposed techniques, our complete framework, Hybrid-Ours, achieves high accuracy at high speed without typical drawbacks. Considering platform-specific optimizations as in Sect. 6.3, our framework, as in Best-version+HF-mode, achieves even higher FPS while maintaining the same level high MOTA. Our method, combining hybrid neural networks using many novel and useful techniques, achieves both state-of-the-art accuracy and speed.

The detection and tracking performance is highly dependent on the detection network. Yolo series used here is less accurate than some networks such as Faster-RCNN, Mask-RCNN, etc. Changing the networks is potential to further improve the performance like MOTA. The key of our method is combining networks with different accuracy-speed trade-offs in a time-interleaving way for both good accuracy and high speed. The LargeNet and SmallNet choices are not limited. The current network switching interval N is 10. Different N is allowed for speed-accuracy trade-offs. A larger N can show larger FPS but lower MOTA. It is also possible to not use any network at some frame to skip the detection and tracking but directly keep the results from the last frame. Such frame skip can be combined with our method to achieve a higher speed while sacrificing accuracy. Moreover, our method is able to mitigate to other platforms than Xavier.

7 Conclusion

We present a novel framework to combine hybrid DNNs with different accuracy-speed trade-offs in a time-multiplexing way for real-time multi-object detection and tracking. By import the tracking knowledge into detection, we allow an inaccurate but fast small DNN for detection in most frames and successfully recover its accuracy loss. By combining different DNNs with space-aware color information, we achieve both state-of-the-art high speed and accuracy. Targeting NVIDIA Xavier, we further optimize the implementation from system and platform level. We achieve 6.8× speedup with similar level high accuracy compared to the traditional large DNN only method on Xavier.

References

1. Abdelali, H.A., et al.: Fast and robust object tracking via accept-reject color histogram-based method. J. Vis. Commun. Image Represent. **34**, 219–229 (2016)
2. Bernardin, K., Stiefelhagen, R.: Evaluating multiple object tracking performance: the CLEAR MOT metrics. J. Image Video Process. **2008**, 1 (2008)
3. Bewley, A., et al.: Alextrac: affinity learning by exploring temporal reinforcement within association chains. In: ICRA, pp. 2212–2218. IEEE (2016)
4. Bewley, A., et al.: Simple online and realtime tracking. In: ICIP, pp. 3464–3468. IEEE (2016)
5. Bochinski, E., et al.: High-speed tracking-by-detection without using image information. In: AVSS, pp. 1–6. IEEE (2017)

6. Bochinski, E., et al.: Extending IOU based multi-object tracking by visual information. In: AVSS, pp. 1–6. IEEE (2018)
7. Danelljan, M., et al.: Adaptive color attributes for real-time visual tracking. In: CVPR, pp. 1090–1097 (2014)
8. Dollár, P., et al.: Fast feature pyramids for object detection. TPAMI **36**(8), 1532–1545 (2014)
9. Hamid Rezatofighi, S., et al.: Joint probabilistic data association revisited. In: ICCV, pp. 3047–3055 (2015)
10. Han, S., et al.: Learning both weights and connections for efficient neural network. In: Advances in Neural Information Processing Systems, pp. 1135–1143 (2015)
11. He, K., et al.: Mask R-CNN. In: ICCV, pp. 2961–2969 (2017)
12. Hubara, I., et al.: Quantized neural networks: training neural networks with low precision weights and activations. J. Mach. Learn. Res. **18**(1), 6869–6898 (2017)
13. Kim, C., et al.: Multiple hypothesis tracking revisited. In: ICCV, pp. 4696–4704 (2015)
14. Leal-Taixé, L., et al.: Motchallenge 2015: towards a benchmark for multi-target tracking. arXiv:1504.01942 (2015)
15. Liu, W., et al.: SSD: single shot MultiBox detector. In: Leibe, B., Matas, J., Sebe, N., Welling, M. (eds.) ECCV 2016. LNCS, vol. 9905, pp. 21–37. Springer, Cham (2016). https://doi.org/10.1007/978-3-319-46448-0_2
16. Possegger, H., et al.: In defense of color-based model-free tracking. In: CVPR, pp. 2113–2120 (2015)
17. Redmon, J., Farhadi, A.: YOLOv3: an incremental improvement. arXiv:1804.02767 (2018)
18. Ren, S., et al.: Faster R-CNN: towards real-time object detection with region proposal networks. In: Advances in Neural Information Processing Systems, pp. 91–99 (2015)
19. Van De Weijer, J., et al.: Learning color names for real-world applications. TIP **18**(7), 1512–1523 (2009)
20. Wojke, N., et al.: Simple online and realtime tracking with a deep association metric. In: ICIP, pp. 3645–3649. IEEE (2017)
21. Womg, A., et al.: Tiny SSD: a tiny single-shot detection deep convolutional neural network for real-time embedded object detection. In: CRV, pp. 95–101. IEEE (2018)
22. Xiang, Y., et al.: Learning to track: Online multi-object tracking by decision making. In: ICCV, pp. 4705–4713 (2015)
23. Yang, B., Nevatia, R.: An online learned CRF model for multi-target tracking. In: CVPR, pp. 2034–2041. IEEE (2012)
24. Yu, F., Li, W., Li, Q., Liu, Y., Shi, X., Yan, J.: POI: multiple object tracking with high performance detection and appearance feature. In: Hua, G., Jégou, H. (eds.) ECCV 2016. LNCS, vol. 9914, pp. 36–42. Springer, Cham (2016). https://doi.org/10.1007/978-3-319-48881-3_3
25. Zhang, L., et al.: Global data association for multi-object tracking using network flows. In: CVPR, pp. 1–8. IEEE (2008)
26. Zhu, G., et al.: MC-HOG correlation tracking with saliency proposal. In: 30th AAAI (2016)