RT-MOT: Confidence-Aware Real-Time Scheduling Framework for Multi-Object Tracking Tasks

Donghwa Kang[†], Seunghoon Lee[‡], Hoon Sung Chwa[§], Seung-Hwan Bae[¶], Chang Mook Kang[‖], Jinkyu Lee[‡] and Hyeongboo Baek[†]

Dept. of Computer Science and Engineering, Incheon National University (INU), Republic of Korea[†]
Dept. of Computer Science and Engineering, Sungkyunkwan University (SKKU), Republic of Korea[‡]
Dept. of Electrical Engineering and Computer Science, DGIST, Republic of Korea[§]
Dept. of Computer Engineering, Inha University, Republic of Korea[¶]
Dept. of Electrical Engineering, Incheon National University (INU), Republic of Korea[‖]
hbbaek@inu.ac.kr

2022 IEEE Real-Time Systems Symposium (RTSS) | 978-1-6654-5346-2/22/$31.00 ©2022 IEEE | DOI: 10.1109/RTSS55097.2022.00035

***Abstract*—Different from existing MOT (Multi-Object Tracking) techniques that usually aim at improving tracking accuracy and average FPS, real-time systems such as autonomous vehicles necessitate new requirements of MOT under limited computing resources: (R1) guarantee of timely execution and (R2) high tracking accuracy. In this paper, we propose RT-MOT, a novel system design for multiple MOT tasks, which addresses R1 and R2. Focusing on multiple choices of a workload pair of *detection* and *association*, which are two main components of the tracking-by-detection approach for MOT, we tailor a measure of object confidence for RT-MOT and develop how to estimate the measure for the next frame of each MOT task. By utilizing the estimation, we make it possible to predict tracking accuracy variation according to different workload pairs to be applied to the next frame of an MOT task. Next, we develop a novel *confidence-aware real-time* scheduling framework, which offers an offline timing guarantee for a set of MOT tasks based on non-preemptive fixed-priority scheduling with the smallest workload pair. At run-time, the framework checks the feasibility of a priority-inversion associated with a larger workload pair, which does not compromise the timing guarantee of every task, and then chooses a feasible scenario that yields the largest tracking accuracy improvement based on the proposed prediction. Our experiment results demonstrate that RT-MOT significantly improves overall tracking accuracy by up to 1.5×, compared to existing popular tracking-by-detection approaches, while guaranteeing timely execution of all MOT tasks.**

I. Introduction

As Multi-Object Tracking (MOT) is essential for various vision applications, there have been many attempts to improve tracking accuracy and average FPS for MOT. In the case of a system integrated with control, if object information is not transmitted according to the control sample period, the controller performs control based on previous data; in the worst case, it can lead to a severe accident in a system such as an autonomous emergency braking system. Therefore, real-time systems such as autonomous vehicles necessitate new requirements of MOT under limited computing resources: (R1) guarantee of timely execution and (R2) high tracking accuracy. Although a recent study in [1] has addressed both R1 and R2 for a single MOT task, no study has achieved both for multiple MOT tasks to be applied to a vision system with multiple cameras.

In this paper, we propose RT-MOT, a novel system design for multiple MOT tasks, which achieves R1 and R2. To this end, we first address the following question.

Q1. How can we design the system architecture of RT-MOT to provide a control knob to explore a trade-off between R1 and R2?

To answer Q1, we focus on the tracking-by-detection structure (one of the most popular structures for MOT) consisting of *detection* of objects in a single video frame and *association* to match objects detected in the current frame with those from previous frames. Based on the structure, RT-MOT implements a new tracking-by-detection structure with multiple choices of a pair of detection and association models. Since different choices for each instance of an MOT task result in different execution times (affecting R1) and different confidence of detected/associated objects (affecting R2), the proposed system architecture addresses Q1, which brings the next question.

Q2. How can we efficiently utilize the proposed system architecture to achieve R1 and R2?

To answer Q2, we need to investigate how R1 and R2 are affected by different pairs of detection and association models selected by an instance of MOT. To address the R2 part, we focus on the reliability of detected/associated objects, and redefine a notion of object confidence for RT-MOT, consisting of motion confidence and appearance confidence, updated by the detection and association parts, respectively. We then develop how to estimate object confidence for the next frame of each MOT, which enables to predict tracking accuracy variation according to different pairs of detection and association models to be applied to the next framework of an MOT task. The prediction is key to exploring a trade-off between R1 and R2 to be utilized in the scheduling framework.

We then develop a novel *confidence-aware real-time* scheduling framework designed for RT-MOT. To this end, we propose a new scheduling algorithm, NPFP$^{\text{flex}}$ (Non-

Hyeongboo Baek is the corresponding author of this paper, and Jinkyu Lee is the co-corresponding author.

2576-3172/22/$31.00 ©2022 IEEE
DOI 10.1109/RTSS55097.2022.00035

Preemptive Fixed-Priority with flexible execution), which addresses the R1 and R2 parts of Q2, respectively by offline/online timing guarantee and the estimation of object confidence. NPFP$^{\text{flex}}$ has the following salient features.

- NPFP$^{\text{flex}}$ offers an offline guarantee of timely execution for all instances of a set of MOT tasks scheduled by non-preemptive fixed-priority scheduling, assuming every instance executes with a pair of detection and association models that requires the shortest execution time. This achieves R1.
- At run-time, NPFP$^{\text{flex}}$ checks the feasibility of execution of another instance (by priority-inversion) with another pair (that may require a longer execution time), without compromising the timely execution of all other instances. Among all feasible scenarios (i.e., which instance and how long to be executed), NPFP$^{\text{flex}}$ chooses the one that yields the largest expected improvement of tracking accuracy based on the estimation of object confidence. This achieves R2 without compromising R1.

We implemented RT-MOT and evaluated its effectiveness in achieving R2 without compromising R1. Our evaluation results show RT-MOT to improve tracking accuracy by up to 1.5× compared to existing popular tracking-by-detection approaches without violating any timing constraint. In addition, we demonstrate that RT-MOT properly selects a pair of detection and association models frame-by-frame, achieving nearly maximum tracking accuracy (achievable without timing constraints) with less total computation.

In summary, this paper makes the following contributions.

- We motivate the importance of choosing a proper pair of detection and association models to explore a trade-off between R1 and R2 (Sec. II).
- We propose the first system design RT-MOT, which addresses R1 and R2 for multiple MOT tasks (outlined in Sec. III).
- We re-define and estimate a measure, which enables to predict tracking accuracy variation to be used in the scheduling framework (Sec. IV).
- We develop a novel confidence-aware real-time scheduling framework for RT-MOT, which offers both offline and online timing guarantees with flexible execution (Sec. V).
- We demonstrate the effectiveness of RT-MOT through experiment on an actual computing system (Sec. VI).

II. Target System and Motivation

This section explains our target system and makes key observations by a measurement-based case study that underlies the design policy of RT-MOT.

A. Target System: DNN-based Multi-Object Tracking

An autonomous vehicle is typically equipped with multiple cameras (e.g., front/side/rear cameras) to sense ambient environments. Each multi-object tracking task is implemented as a *periodic task* that takes a video frame from a camera and performs DNN-based computation to predict trajectories of multiple targets in frame sequences; this process is repeated periodically. Timely response with high tracking accuracy is a necessity because each task has a deadline, and its result is used as input for other components (i.e., motion planning) of the car.

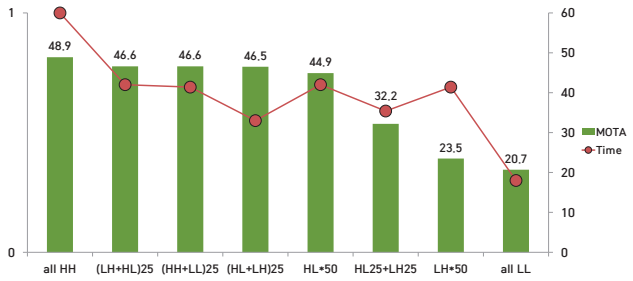

Fig. 1. Tracking accuracy (by a well-known measure MOTA) and execution time (normalized by the largest one) under different combinations of detection and association schemes over 50 consecutive frames

For multi-object tracking, the tracking-by-detection approach [2], [3] is one of the most popular ones, and it is characterized by two main steps: 1) *detection* of objects in a single video frame and 2) *association* to match objects detected in the current frame with those from previous frames. Thus, its tracking process can be divided into a front-end detector followed by a back-end tracker. Since object detection is a standalone step in the tracking process, one benefit is the flexibility to pair different object detection models with different association strategies.

B. Trade-off between Execution Time and Tracking Accuracy

Although the tracking-by-detection approach offers high flexibility in utilizing various object detection models and association methods, it is still very challenging to achieve both accurate tracking and timely response on resource-constrained computing platforms since the execution time and accuracy often conflict with each other. We present a case study to investigate the effect of applying different detection models and association methods on execution time and tracking accuracy and identify the main challenges faced therein.

In the case study, we employ YOLOv5[1] with two different inputs: i) processing a full-size frame (referred to as *high-confidence detection*) and ii) processing a partial portion of a frame (referred to as *low-confidence detection*); the details will be explained in Sec. III-B. We also employ two different popular association strategies: i) performing both feature- and Intersection-over-Union (IoU)-based methods (referred to as *high-confidence association*) and ii) performing the IoU-based method only (referred to as *low-confidence association*).

Fig. 1 shows the effect of applying 6 different combinations of a pair of detection and association schemes over 50 consecutive video frames on the execution time and tracking accuracy. In the figure, the x-axis represents different combinations; ($\mathtt{D}^x$,$\mathtt{A}^y$) expresses a choice of detection and association schemes for a single frame, where x and y can

[1]https://github.com/ultralytics/yolov5

be either H (high-confidence) or L (low-confidence). For example, $((D^H,A^H)+(D^L,A^L))\cdot 25$ denoted by (HH+LL)·25 refers 25 repetitions of two consecutive frames, which perform a combination of high-confidence detection and high-confidence association, and then that of low-confidence detection and low-confidence association. We consider meaningful combinations whose H (as well as L) ratio for detection/association is exactly 50%, including 50 identical frames, 25 identical chunks each consisting of two different frames, and 25 identical frames followed by another 25 identical frames. In addition, we add all HH and all LL for reference.

We summarize the following observations from Fig. 1.

O1. There exists a trade-off between execution time and tracking accuracy in choosing the ratio of high-confidence detection/association.

A higher ratio of high-confidence detection/association achieves higher tracking accuracy at the expense of more execution time. For example, (HH) improves the tracking accuracy by 48.9% but requires more execution time by 3.3× over (LL). Note that we use MOTA (Multiple Object Tracking Accuracy), one of the widely used object tracking accuracy metrics, for accuracy measurement, which will be discussed in Sec. VI. Also, note that the execution time shown in Fig. 1 is the total sum of the execution times of all frames and is normalized to the case of (HH).

O2. Tracking accuracy varies greatly with different combinations of a choice of detection and association schemes, although the combinations yield similar computation time.

In Fig. 1, all combinations except (HH) and (LL) have the same ratio (i.e., 50%) of high-confidence detection and association applied to 50 consecutive video frames, resulting in comparable execution times. On the other hand, tracking accuracy varies widely up to a difference of 14.4 percentage points (%p) depending on not only the choice of detection and association schemes for each frame but also the collection of choices for consecutive frames. For example, (HH+LL)·25 and HL·25+LH·25 shows 46.6% and 32.2% of tracking accuracy, respectively (meaning that the former yields 44.7% higher accuracy than the former), while their total computation times differ only by 16.4%.

Observation O1 and O2 give rise to opportunities and challenges *together* in achieving both timely response and maximum tracking accuracy under limited computing resources. By O1, we can *dynamically* decide which detection and association schemes are to be applied for each frame of a multi-object tracking task by trading off the execution cost for accuracy. If there is less spare time before the next frame arrives, a multi-object tracking task can select low-confidence detection and/or association to finish its execution before the deadline at the expense of sacrificing tracking accuracy. On the other hand, if there is enough spare time before the next frame arrives, a multi-object tracking task can select both high-confidence detection and association to improve tracking accuracy.

By O2, it is nevertheless difficult to find an optimal combination of a choice of detection and association schemes for each task that maximizes tracking accuracy without compromising the timely execution of the task. This is because i) there exists a huge number of possible combinations of the choices of detection and association schemes for a frame sequence, and ii) the tracking accuracy of one combination also dynamically varies with the input frame sequence. Moreover, when multiple tracking tasks are scheduled together by sharing limited computing resources, a selected combination of detection and association schemes for a task can affect not only the timely execution and tracking accuracy of the task itself, but also those of the other tasks. This makes it very challenging to find the right solution.

III. RT-MOT: System Design

In this section, we present our goal and design of RT-MOT, based on O1 and O2 in Sec. II.

A. System Goal and Overview

We aim to develop an online tracking-accuracy-aware scheduling framework, which achieves the following goals for real-time multi-object tracking tasks:

R1. It provides timing guarantees for real-time multi-object tracking tasks; and

R2. It maximizes overall tracking accuracy.

To this end, we propose a new system abstraction, named RT-MOT, that enables frame-level flexible scheduling of multi-object tracking tasks. In particular, RT-MOT supports dynamic selection of different execution models for detection and association, respectively, for each frame; it makes run-time frame-level scheduling decisions by considering the effect of execution model selection on both timely execution and tracking accuracy of multi-object tracking tasks. We address the following key issues, which enables RT-MOT to maximize overall tracking accuracy while meeting all timing constraints:

I1. How to estimate the variation of overall accuracy according to different detector/tracker selections of the next frame?

I2. How to provide an offline timing guarantee to multiple MOT tasks while maximizing the overall accuracy at run-time using the answer of I1?

I3. How to design the system architecture that supports flexible tracking-by-detection and provides an interface that can accommodate the answer of I1 and I2?

B. Design of RT-MOT

RT-MOT supports frame-level flexible scheduling for multi-object tracking tasks to maximize overall tracking accuracy while providing real-time guarantees. As depicted in Fig. 2, the core design features of RT-MOT include dynamic tracking-by-detection execution pipeline (addressing I3), multi-object tracking confidence estimator (addressing I1 and to be detailed in Sec. IV), and frame-level flexible scheduler (addressing I2 and to be detailed in Sec. V) as follows.

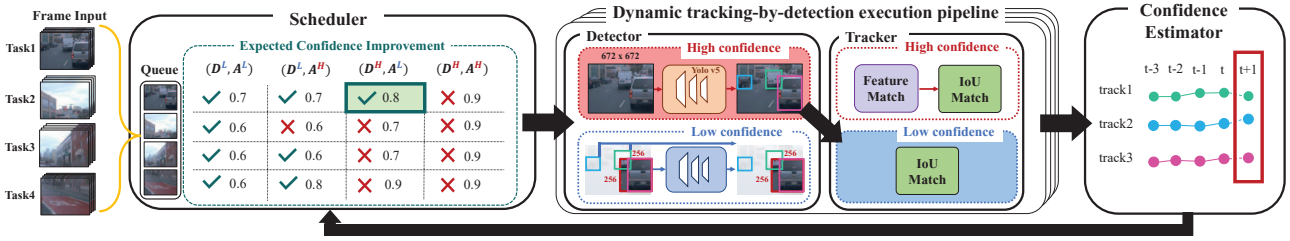

Fig. 2. Overall system design of RT-MOT

Dynamic tracking-by-detection execution pipeline. RT-MOT implements a new tracking-by-detection approach that is capable of combining different front-end detectors with different back-end trackers at frame by frame, each of which exhibits different execution costs and tracking confidence.

Under RT-MOT, each input frame of a multi-object tracking task is forwarded to its corresponding tracking-by-detection execution pipeline. The front-end detector identifies the location and size of each object's bounding box in the input frame and sends the detection information to the back-end tracker. The back-end tracker then associates each detected object with one of the existing tracks from previous frames (i.e., tracklets) based on motion and/or appearance similarity and updates the tracking information of the matched tracklets.

We consider two types of execution model for detection and association, respectively: *high-confidence* and *low-confidence* execution models as shown in Fig. 2. The high-confidence execution models exhibit higher tracking confidence at the expense of more execution time, while the low-confidence execution models consume less execution time at the expense of sacrificing tracking confidence.

The tracking-by-detection execution pipeline uses YOLOv5 as a front-end detector. It can accept variable frame sizes as input, and its inference latency depends on the input size. The front-end detector employs YOLOv5 with two different inputs: i) processing an entire frame with the size of 672×672 (referred to as *high-confidence detection*) and ii) processing a partial portion of a frame only containing objects of interest with the size of 256×256 (referred to as *low-confidence detection*). For low-confidence detection, we divide an input frame into smaller portions with the size of 256×256, extract a particular frame portion by image cropping, and use the cropped image as input for YOLOv5. Processing a cropped portion with a smaller size can effectively lower the computational workload with no detection accuracy drop in the cropped portion [4]. Note that the particular portion of a frame will be henceforth referred to as *Region-of-Interest* (RoI). Among the frame portions as large as RoI, we select the one whose average confidence score of the tracklets therein is the lowest, which efficiently improves tracking accuracy, where confidence score will be explained in Sec. IV. To handle the rest portions outside of RoI, a set of tracks outside of RoI (obtained by the previous frame) is also included in the result of detection to keep maintained by the back-end tracker.

The back-end tracker adopts two popular association methods: i) performing both feature- and Intersection-over-Union (IoU)-based methods (referred to as *high-confidence association*) and ii) performing the IoU-based method only (referred to as *low-confidence association*). The IoU-based method [3] is known as the simplest form of object association. The feature-based method [2] is one of the advanced association methods to handle tracking loss due to occlusion between objects. It incorporates a re-identification model as a feature extractor to extract the feature vectors of the detected objects and uses the feature vectors to match already confirmed targets against new detected objects to re-identify occluded targets that are temporally lost, which involves extra computational cost. Therefore, our proposed tracking-by-detection execution pipeline has four (2×2) different choices of a pair of detection and association models in total; each pair is denoted as $(\mathtt{D}^x, \mathtt{A}^y)$, where x and y can be either `H` (high-confidence) or `L` (low-confidence). Among them, the scheduler dynamically chooses one pair to be processed for each frame of a multi-object tracking task.

Multi-object tracking confidence estimator. We define a novel multi-object tracking confidence metric to gauge the confidence level of a list of tracklets for each multi-object tracking task. Using this metric, the confidence estimator estimates how the confidence level will vary depending on the choice of a pair of detection and association models to process the next frame of a task.

After each task finishes its tracking-by-detection execution pipeline, the confidence estimator evaluates the confidence of each tracklet in consideration of the length and continuity of a tracklet and the similarity with the associated detection (see Sec. IV-A for details). In general, a tracklet has a higher confidence score if it is frequently and recently associated with a detected object having a strong affinity. The confidence estimator then predicts the amount of potential increase in each tracklet's confidence score depending on the choices of a pair of detection and association models for the next frame of a task (see Sec. IV-B for details). Such a predicted decrease in the confidence score is utilized by the scheduler to determine a pair of detection and association models for each frame and a schedule for tasks.

Frame-level flexible scheduler. RT-MOT implements an online scheduler that not only dynamically determines a pair of detector and tracker for each frame but also adaptively

generates a flexible schedule by considering confidence estimates for tasks to maximize overall tracking accuracy without violating any timing constraints at run-time.

The scheduler runs as a background daemon to communicate with the tracking-by-detection execution pipelines through the IPC-based communication stub interface implemented within each task and the scheduler. The scheduler is invoked upon arrival of a new frame for each task, or completion of a task. Based on each task's static model parameters (in Sec. V-A), the scheduler determines each frame's priority and its pair of detector and tracker according to non-preemptive fixed-priority with the minimum execution by default (in Sec. V-B). However, upon each invocation, the scheduler checks the possibility of execution of a frame (regardless of its priority) with a pair of detector and tracker that yields the maximum overall tracking accuracy without violating any timing constraints guaranteed by the offline schedulability analysis (in Sec. V-C).

IV. Tracking Confidence Estimation

This section describes the notion of tracklet confidence tailored to the proposed dynamic tracking-by-detection execution pipeline and presents a tracklet confidence estimation method that plays a key role in pair selection and scheduling decisions by the scheduling framework of RT-MOT.

A. Tracklet Confidence

We employ the notion of *tracklet confidence* to measure the reliability of a tracklet constructed during object tracking over time, which is widely used in the field of multi-object tracking; a tracklet is generated by the frame-by-frame association based on the tracking-by-detection approach [5], [6]. We tailor the notion of tracklet confidence to the proposed dynamic tracking-by-detection execution pipeline and explain how to calculate the tracklet confidence score.

For a video frame, let O_i^t denote as the i-th object response detected at the t-th frame, and it is characterized by (M_i^t, A_i^t), where M_i^t and A_i^t are the *motion* and *appearance* states of O_i^t, respectively. The motion state M_i^t is further specified as $M_i^t = (p_i^t, s_i^t, v_i^t)$, where $p_i^t = (x_i^t, y_i^t)$, $s_i^t = (w_i^t, h_i^t)$, $v_i^t = (vx_i^t, vy_i^t, vw_i^t, vh_i^t)$ are the position, size, and velocity of O_i^t, respectively. The appearance state A_i^t is the feature vector of O_i^t extracted by a feature extractor. Note that M_i^t and A_i^t can be obtained from the results of the front-end detector and back-end tracker, respectively, in the tracking-by-detection execution pipeline. We then define a tracklet χ_i^t of object O_i^t as a set of tracks followed by O_i^t up to the t-th frame, and it is expressed as $\chi_i^t = \{O_i^k | 1 \le t_i^s \le k \le t_i^e \le t\}$, where t_i^s and t_i^e are the start- and end-frame of the tracklet. A set of tracklets of all objects up to the t-th frame is denoted as $\Phi_{1:t}$.

We now explain how to calculate the tracklet confidence score. A tracklet with a high confidence score is considered as a reliable tracklet, while another tracklet with a low confidence score is considered as an unreliable tracklet with a fragmented trajectory due to inaccurate detection and/or association. We model tracklet confidence $\Omega(\chi_i^t)$ of χ_i^t as $\Omega(\chi_i^t) = \Omega_M(M_i^t) \times \Omega_A(A_i^t)$, where $\Omega_M(M_i^t)$ and $\Omega_A(A_i^t)$ are motion and appearance confidence of χ_i^t, respectively, each of whose range is [0,1].

Under RT-MOT, after each task finishes its tracking-by-detection execution pipeline for the t-th frame, the states of each tracklet $\chi_i^t \in \Phi_{1:t}$ are updated as well as its confidence score. Depending on the result of the association between a set of tracklets and a set of detected objects at the t-th frame, each tracklet $\chi_i^t \in \Phi_{1:t}$ can be classified into three categories:

- CG1. χ_i^t is matched with one of the detected objects by *high-confidence* association;
- CG2. χ_i^t is matched with one of the detected objects by *low-confidence* association; and
- CG3. χ_i^t is unmatched with any of the detected objects.

Note that the detected objects shown in CG1–CG3 are the results of either high- or low-confidence detection. Then, the motion and appearance confidence scores of the tracklet χ_i^t, which corresponds to one of the CG1, CG2, and CG3 cases, can be updated as follow:

- $\Omega_M(M_i^t) = \Omega_A(A_i^t) = 1$, if χ_i^t belongs to CG1; (1a)
- $\Omega_M(M_i^t) = 1$ and $\Omega_A(A_i^t) = \left[\Omega_A(A_i^{t\text{-}1}) \times \Delta A_i^{t\text{-}1}\right]_0$, if χ_i^t belongs to CG2; (1b)
- $\Omega_M(M_i^t) = \left[\Omega_M(M_i^{t\text{-}1}) \times \Delta M_i^{t\text{-}1}\right]_0$ and $\Omega_A(A_i^t) = \left[\Omega_A(A_i^{t\text{-}1}) \times \Delta A_i^{t\text{-}1}\right]_0$, if χ_i^t belongs to CG3. (1c)

where $[X]_Y := \max(X, Y)$, and we denote by $\Delta M_i^{t\text{-}1}$ and $\Delta A_i^{t\text{-}1}$ the amounts of variations of the motion and appearance states ($M_i^{t\text{-}1}$ and $A_i^{t\text{-}1}$) at the (t-1)-th frame from the most-recently-updated motion and appearance states before the (t-1)-th frame (denoted as $M_i^{t\text{-}f}$ and $A_i^{t\text{-}g}$), respectively, where $f, g > 1$. We can calculate $\Delta M_i^{t\text{-}1}$ as

$$\Delta M_i^{t\text{-}1} = \Lambda_s(M_i^{t\text{-}f}, M_i^{t\text{-}1}) \times \Lambda_v(M_i^{t\text{-}f}, M_i^{t\text{-}1}), \tag{2}$$

where

$$\Lambda_s(M_i^{t\text{-}f}, M_i^{t\text{-}1}) = -\frac{1}{4}\left(\frac{h_i^{t\text{-}f} - h_i^{t\text{-}1}}{h_i^{t\text{-}f} + h_i^{t\text{-}1}} + \frac{w_i^{t\text{-}f} - w_i^{t\text{-}1}}{w_i^{t\text{-}f} + w_i^{t\text{-}1}}\right) + \frac{1}{2}, \tag{3}$$

$$\Lambda_v(M_i^{t\text{-}f}, M_i^{t\text{-}1}) = 1 - 2\left|\sigma\left(\frac{vx_i^{t\text{-}f} - vx_i^{t\text{-}1}}{vx_i^{t\text{-}f} + vx_i^{t\text{-}1}} + \frac{vy_i^{t\text{-}f} - vy_i^{t\text{-}1}}{vy_i^{t\text{-}f} + vy_i^{t\text{-}1}}\right) - \frac{1}{2}\right|, \tag{4}$$

and σ is a sigmoid function [7], [8]. We also calculate $\Delta A_i^{t\text{-}1}$ as

$$\Delta A_i^{t\text{-}1} = \Lambda_a(A_i^{t\text{-}g}, A_i^{t\text{-}1}) = \frac{(A_i^{t\text{-}g} \cdot A_i^{t\text{-}1})}{|A_i^{t\text{-}g}||A_i^{t\text{-}1}|}. \tag{5}$$

The terms $\Lambda_s(M_i^{t\text{-}f}, M_i^{t\text{-}1})$, $\Lambda_v(M_i^{t\text{-}f}, M_i^{t\text{-}1})$, and $\Lambda_a(A_i^{t\text{-}g}, A_i^{t\text{-}1})$ shown in Eqs. (3)–(5) measure the amounts of variations of size, velocity, and appearance, respectively, each of whose range is in [0,1].[2] Intuitively, if the size (*likewise* acceleration/deceleration) of a tracklet becomes smaller

[2]Those terms are widely used in other object tracking papers, e.g., [9], as a form of exponential functions. For simplicity, we normalize the exponential functions to be in the range of [0,1].

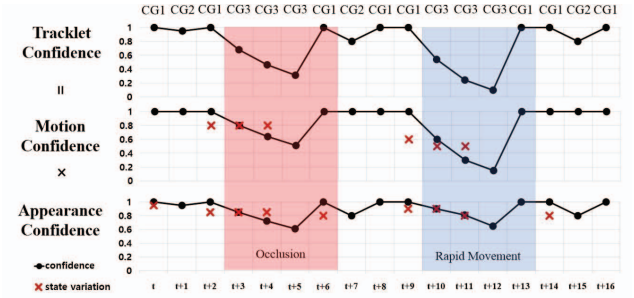

Fig. 3. An example of a series of tracklet confidence change according to each frame's situation among CG1–CG3

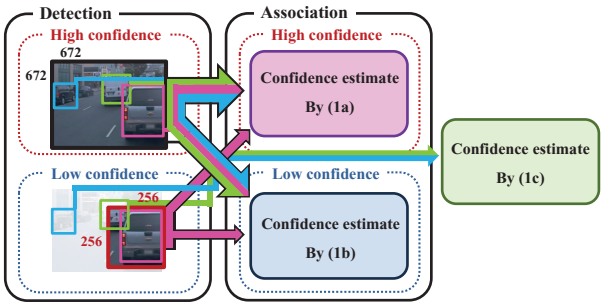

Fig. 4. An illustration of how each tracket calculates its confidence estimate according to different choices of a pair of detection and association models

(*likewise* larger) from the (t-f)-th to (t-1)-th frame, the value of $\Lambda_s(M_i^{t-f}, M_i^{t-1})$ (*likewise* $\Lambda_v(M_i^{t-f}, M_i^{t-1})$) is close to 0, yielding a large decrease in the motion confidence score of the tracklet if unmatched at the t-th frame. Similarly, if there exists a large dissimilarity between the feature vectors of the tracks at the (t-g)-th and (t-1)-th frame due to occlusion, the value of $\Lambda_a(A_i^{t-g}, A_i^{t-1})$ is close to 0, yielding a large decrease in the appearance confidence score of the tracklet if unmatched at the t-th frame. For the newly detected objects that are not associated with any of the existing tracklets in $\Phi_{1:t}$, we create a new tracklet for them to track their trajectories from the t-th frame.

Fig. 3 demonstrates how tracklet confidence (accordingly with its motion and appearance confidence) varies depending on CG1–CG3. From frame (t+3) to frame (t+5), a target object is temporarily occluded by other objects, so its corresponding tracklet is unmatched (belonging to CG3). Therefore, Eq. (1c) updates its motion and appearance confidence scores (represented by black dots in the figure). As seen in Eq. (1c), the motion confidence score of the current frame is calculated by multiplication of that of the previous frame and its state variation (represented by red crosses in the figure); considering the state variation in $[0, 1]$, the motion confidence score of the current frame decreases. The same holds for the appearance confidence scores as shown in Eq. (1c). Therefore, the tracklet confidence score (i.e., multiplication of the motion and appearance confidence scores) decreases from 1 at frame (t+2) to 0.31 at frame (t+5). In frame (t+6), the occluded object is correctly re-identified by high-confidence association (belonging to CG1). Then, the tracklet confidence score is set to 1. A similar trend can be seen from frame (t+10) to frame (t+12) when the target object exhibits rapid movement with high acceleration. Meanwhile, the tracklet exhibits a large variation in its motion state, yielding a larger decrease in its motion confidence score so does the tracklet confidence score.

B. Tracklet Confidence Prediction for the Next Frame

We now present a method to predict tracklet confidence depending on different choices of a pair of detection and association models for the next frame, which will play a key role in pair selection and scheduling decision by the scheduling framework (to be discussed in Sec. V) to improve overall tracking accuracy. For a given set of tracklets $\Phi_{1:t}$ up to the t-th frame, we construct $\Phi_{1:t+1}$ and calculate a tracklet confidence estimate $\overline{\Omega}(\chi_i^{t+1})$ at the (t+1)-th frame based on the following assumptions, for four cases of ($\mathtt{D}^{\mathtt{H}}$,$\mathtt{A}^{\mathtt{H}}$), ($\mathtt{D}^{\mathtt{L}}$,$\mathtt{A}^{\mathtt{H}}$), ($\mathtt{D}^{\mathtt{H}}$,$\mathtt{A}^{\mathtt{L}}$), and ($\mathtt{D}^{\mathtt{L}}$,$\mathtt{A}^{\mathtt{L}}$) as shown in Fig. 4:

- In case of ($\mathtt{D}^{\mathtt{H}}$,$\mathtt{A}^{\mathtt{H}}$), all tracklets in $\Phi_{1:t}$ belong to CG1;
- In case of ($\mathtt{D}^{\mathtt{L}}$,$\mathtt{A}^{\mathtt{H}}$), a subset of tracklets in RoI belongs to CG1, and the rest of tracklets outside RoI belongs to CG3;
- In case of ($\mathtt{D}^{\mathtt{H}}$,$\mathtt{A}^{\mathtt{L}}$), all tracklets in $\Phi_{1:t}$ belong to CG2;
- In case of ($\mathtt{D}^{\mathtt{L}}$,$\mathtt{A}^{\mathtt{L}}$), a subset of tracklets in RoI belongs to CG2, and the rest of tracklets outside RoI belongs to CG3.

Tracklet confidence estimate $\overline{\Omega}(\chi_i^{t+1})$ at the (t+1)-th frame is then calculated by using Eqs. (1a)–(1c) in accordance with the category of χ_i^{t+1}. Finally, we define the *expected* confidence score $\overline{\mathbf{\Omega}}_{\tau_k}(\Phi_{1:t+1}, (\mathtt{D}^x, \mathtt{A}^y))$ of a MOT task τ_k for the choice of ($\mathtt{D}^x$,$\mathtt{A}^y$), where x, $y \in \{\mathtt{H}, \mathtt{L}\}$, at the ($t$+1)-th frame as

$$\overline{\mathbf{\Omega}}_{\tau_k}(\Phi_{1:t+1}, (\mathtt{D}^x, \mathtt{A}^y)) = \frac{\sum_{\forall \chi_i^{t+1} \in \Phi_{1:t+1}} \overline{\Omega}(\chi_i^{t+1})}{|\Phi_{1:t+1}|}, \quad (6)$$

where $|\Phi_{1:t+1}|$ is the total number of tracklets at the (t+1)-th frame. We also define the *measured* confidence score $\mathbf{\Omega}_{\tau_k}(\Phi_{1:t})$ of a task τ_k at the t-th frame as

$$\mathbf{\Omega}_{\tau_k}(\Phi_{1:t}) = \frac{\sum_{\forall \chi_i^t \in \Phi_{1:t}} \Omega(\chi_i^t)}{|\Phi_{1:t}|}. \quad (7)$$

Using the measure, the amount of *expected* increase in τ_k's confidence score for the choice of ($\mathtt{D}^x$,$\mathtt{A}^y$) at the (t+1)-th frame (denoted as $\Delta\overline{\mathbf{\Omega}}_{\tau_k}(\Phi_{1:t+1}, (\mathtt{D}^x, \mathtt{A}^y))$) can be calculated as

$$\Delta\overline{\mathbf{\Omega}}_{\tau_k}(\Phi_{1:t+1}, (\mathtt{D}^x, \mathtt{A}^y)) = \overline{\mathbf{\Omega}}_{\tau_k}(\Phi_{1:t+1}, (\mathtt{D}^x, \mathtt{A}^y)) - \mathbf{\Omega}_{\tau_k}(\Phi_{1:t}). \quad (8)$$

For simplicity of presentation, we will use the notation of $\Delta\overline{\mathbf{\Omega}}_{\tau_k}$, for $\Delta\overline{\mathbf{\Omega}}_{\tau_k}(\Phi_{1:t+1}, (\mathtt{D}^x, \mathtt{A}^y))$. By deriving $\Delta\overline{\mathbf{\Omega}}_{\tau_k}$, we establish a reasonable criterion to determine which MOT task's frame should be executed among multiple MOT tasks, to be utilized by the scheduler. That is, if we focus on improving

overall tracking accuracy *only*, the scheduler chooses the MOT task with the largest $\Delta\overline{\mathbf{\Omega}}_{\tau_k}$ (that yields the largest confidence score improvement). However, the scheduler also considers the timing guarantees of multiple MOT tasks, to be addressed in the following section.

V. Scheduling Framework for RT-MOT

In this section, we consider NPFP$^{\text{min}}$, a traditional non-preemptive fixed-priority scheduling in which every instance (i.e., job) of a set of multi-object tracking tasks executes for the minimum execution requirement by selecting ($\mathtt{D}^{\mathtt{L}}$,$\mathtt{A}^{\mathtt{L}}$). We then develop a novel scheduling framework designed for RT-MOT, called NPFP$^{\text{flex}}$ (Non-Preemptive Fixed-Priority with flexible execution), so as to achieve two important design principles.

- First, NPFP$^{\text{flex}}$ shares the existing offline schedulability test for NPFP$^{\text{min}}$ (to be presented in Lemma 1), which offers timely execution of every instance (i.e. job) of a set of multi-object tracking tasks.
- Second, NPFP$^{\text{flex}}$ checks the feasibility for each active job to be executed (regardless of its priority by FP) beyond its minimum execution requirement, without compromising the schedulability of any future jobs to executed according to NPFP$^{\text{min}}$. Among the active jobs that do not compromise the schedulability, NPFP$^{\text{flex}}$ chooses to execute the job that yields the largest expected improvement of confidence score among all pairs of a task and ($\mathtt{D}^{\mathtt{L/H}}$,$\mathtt{A}^{\mathtt{L/H}}$) from Eq. (8), which in turn improves overall accuracy of MOT.

While it is difficult to embrace the second principle only, it is more challenging to establish the two principles together. In this section, we address this real-time scheduling problem.

A. Task Model

To model multi-object tracking tasks that utilize RT-MOT, we use a strictly periodic task model [10]. A task $\tau_i \in \tau$ is specified by (T_i, C_i), where T_i is the period and C_i is the worst-case execution time (WCET). A task τ_i invokes a series of jobs every T_i times; once a job is released at t_0, it should finish its execution no later than $t_0 + T_i$. A job is said to be *active* at t_0, if it is released no later than t_0 and has remaining execution at t_0. Considering the non-preemptiveness of each job (executed on GPU without preemption), a job does not pause before completion, once it starts to execute. We consider uniprocessor scheduling in which only a single job can be executed on the computing platform at any time; this accords with our system architecture in which every instance of a MOT task monopolizes GPU when it is executed.

The WCET C_i can be decomposed by C_i^D and C_i^A, i.e., $C_i = C_i^D + C_i^A$. The former is for the detection part, while the latter is for the association part.

In the detection part, $c_i^{infer}(\mathtt{L}$ or $\mathtt{H})$ is the total inference time for YOLOv5 that has the two different WCETs for low-confidence ($\mathtt{L}$) and high-confidence ($\mathtt{H}$) detection. Then, C_i^D can be calculated as follows: $C_i^D(\mathtt{L}$ or $\mathtt{H}) = c^{pre} + c_i^{infer}(\mathtt{L}$ or $\mathtt{H})$, where c^{pre} denotes the WCET for RoI identification and image cropping.

In the association part, $c_i^{as}(\mathtt{L}$ or $\mathtt{H})$ is the WCET for low-confidence ($\mathtt{L}$) and high-confidence ($\mathtt{H}$) association. As explained in Sec. III-B, the low-confidence association performs a simple IoU-based matching algorithm (whose WCET is denoted by c_i^{IoU}), while high-confidence association *additionally* performs a feature-based method (whose WCET is denoted by $c_i^{cascade}$). Therefore, $c_i^{as}(\mathtt{L}) = c_i^{IoU}$ and $c_i^{as}(\mathtt{H}) = c_i^{cascade} + c_i^{IoU}$ hold, which implies $c_i^{as}(\mathtt{L}) < c_i^{as}(\mathtt{H})$. Then, we can express C_i^A as follows: $C_i^A(\mathtt{L}$ or $\mathtt{H}) = c_i^{as}(\mathtt{L}$ or $\mathtt{H}) + c^{post}$, where c^{post} denotes the WCET for updating the confidence of all tracklets in each task.

Since we have two options (i.e., $\mathtt{L}$ and $\mathtt{H}$) for each of the detection and association parts, we have four options of C_i (i.e., the WCET of τ_i), where the first and second characters in the superscript of C_i (each of which is either $\mathtt{L}$ or $\mathtt{H}$) correspond to the detection and association parts, respectively.

- $C_i^{\mathtt{LL}} = C_i^D(\mathtt{L}) + C_i^A(\mathtt{L})$: the WCET of τ_i for ($\mathtt{D}^{\mathtt{L}}$,$\mathtt{A}^{\mathtt{L}}$)
- $C_i^{\mathtt{HL}} = C_i^D(\mathtt{H}) + C_i^A(\mathtt{L})$: the WCET of τ_i for ($\mathtt{D}^{\mathtt{H}}$,$\mathtt{A}^{\mathtt{L}}$)
- $C_i^{\mathtt{LH}} = C_i^D(\mathtt{L}) + C_i^A(\mathtt{H})$: the WCET of τ_i for ($\mathtt{D}^{\mathtt{L}}$,$\mathtt{A}^{\mathtt{H}}$)
- $C_i^{\mathtt{HH}} = C_i^D(\mathtt{H}) + C_i^A(\mathtt{H})$: the WCET of τ_i for ($\mathtt{D}^{\mathtt{H}}$,$\mathtt{A}^{\mathtt{H}}$)

In this section, we employ FP (Fixed-Priority) scheduling in which each task has a static priority and each job inherits the priority of its invoking task. Let $\mathsf{HP}(\tau_i)$ and $\mathsf{LP}(\tau_i)$ respectively denote a set of tasks whose priority is higher than τ_i and lower than τ_i. The *response time* of a job of τ_i is defined as the duration between the release and completion of the job. A *level-i busy period* is defined as the longest consecutive time interval during which the computing unit is occupied by jobs whose priority is higher than or equal to τ_i. Let LHS and RHS denote the left-hand-side and right-hand-side, respectively.

B. NPFP$^{\text{min}}$: Base Scheduling Algorithm

To develop NPFP$^{\text{flex}}$, we first explain its base scheduling algorithm NPFP$^{\text{min}}$, which is the same as the traditional non-preemptive fixed-priority scheduling with $C_i = C_i^{\mathtt{LL}}$ for every $\tau_i \in \tau$. We focus on t_0, at which at least one job is released when the computing platform is idle, or a job finishes its execution. At every t_0, we choose the job of τ_i, whose priority (inherited by its invoking task) is the highest among the jobs active at t_0. We then execute the chosen job for ($\mathtt{D}^{\mathtt{L}}$,$\mathtt{A}^{\mathtt{L}}$) during at most $C_i^{\mathtt{LL}}$.

Then, we present an existing offline schedulability test for NPFP$^{\text{min}}$ in the following lemma.

Lemma 1 (In [11]–[13]): Suppose that a task set τ is scheduled by the NPFP$^{\text{min}}$ scheduling algorithm. If every task $\tau_i \in \tau$ satisfies Eq. (9), every job invoked by tasks in τ cannot miss its deadline.

$$R_i \leq T_i, \tag{9}$$

where R_i, the worst-case response time of τ_i, is calculated by finding $R_i(x+1) = R_i(x)$ through iteration from $R_i(0) = C_i^{\mathtt{LL}} + max_{\tau_j \in \mathsf{LP}(\tau_i)} C_j^{\mathtt{LL}}$ in Eq. (10).

$$R_i(x+1) = C_i^{\mathtt{LL}} + \max_{\tau_j \in \mathsf{LP}(\tau_i)} C_j^{\mathtt{LL}} + \sum_{\tau_h \in \mathsf{HP}(\tau_i)} \left\lceil \frac{R_i(x)}{T_h} \right\rceil \cdot C_h^{\mathtt{LL}} \tag{10}$$

Proof: Without loss of generality, let $t = 0$ be the release time of the job of τ_i (denoted by J_i). Lemma 6 in [14] proves that the worst-case response time of τ_i (which is not necessarily from J_i, but can be from the following job of τ_i) is found in a level-i busy period when all higher-priority jobs are released at $t = 0$ and a lower-priority job whose execution time is the largest is released right before $t = 0$. In an interval of length L that starts at $t = 0$, the amount of execution of all jobs belonging to the former is upper-bounded by $\sum_{\tau_h \in \mathsf{HP}(\tau_i)} \lceil \frac{L}{T_h} \rceil \cdot C_h^{\mathtt{LL}}$, while the amount of the job that coincides with the latter is upper-bounded by $\max_{\tau_j \in \mathsf{LP}(\tau_i)} C_j^{\mathtt{LL}}$. Therefore, the RHS of Eq. (10) is the sum of the WCET of J_i (i.e., $C_i^{\mathtt{LL}}$) and that of all jobs executed before J_i's execution. Therefore, if Eq. (9) holds for τ_i, the earliest job of τ_i (i.e., J_i) in any level-i busy period does not miss its deadline if it executes for up to $C_i^{\mathtt{LL}}$.

In the second part of the proof, we prove $t_{x+1} - t_x \leq T_i$, where t_x and t_{x+1} respectively denote the time instants at which the x^{th} and $(x+1)^{th}$ earliest jobs of τ_i start their execution in the same level-i busy period; the proof is similar to Lemma 2 of [12] that shows no self-pushing phenomenon under some conditions. At t_x, there is no higher-priority active job; otherwise, the x^{th} job cannot start its execution at t_x. Therefore, in an interval of length L that starts at t_x, the amount of executions of the x^{th} job of τ_i and other jobs whose priority is higher than τ_i is $C_i^{\mathtt{LL}} + \sum_{\tau_h \in \mathsf{HP}(\tau_i)} \lceil \frac{L}{T_h} \rceil \cdot C_h^{\mathtt{LL}}$ (denoted by $R'_i(L)$). We can confirm that $R'_i(L)$ is no larger than the RHS of Eq. (10) with $R_i(x) = L$. Therefore, the supposition (i.e., τ_i satisfies Eq. (9)) implies that there exists $L = R'_i(L)$ that satisfies $R'_i(L) \leq T_i$, meaning that the $(x+1)^{th}$ job of τ_i can start its execution no later than T_i time units after t_x.

The first and second parts of the proof respectively operate as the base and inductive cases for mathematical induction, which proves the lemma. ■

C. NPFP$^{\mathsf{flex}}$: Novel Scheduling Framework for RT-MOT

We now develop NPFP$^{\mathsf{flex}}$ in Algorithm 1, designed for RT-MOT. In terms of the job prioritization and the job execution requirement, NPFP$^{\mathsf{flex}}$ by default follows the policy of NPFP$^{\mathsf{min}}$ by setting flex to F (Line 1), at every t_0 where at least a job is released when the computing platform is idle, or a job finishes its execution. Then, we check the feasibility for each active job (denoted by J_k of τ_k in Line 2) to be executed for the given execution requirement C_k (either $C_k^{\mathtt{LL}}$, $C_k^{\mathtt{LH}}$, $C_k^{\mathtt{HL}}$, or $C_k^{\mathtt{HH}}$ in Line 3). To guarantee the feasibility of a job, we need to guarantee (a) no deadline miss of J_k if it starts its execution for C_k at t_0, *and* (b) no deadline miss of all future jobs to be executed after J_k according to NPFP$^{\mathsf{min}}$ (i.e., according to FP with $C_i = C_i^{\mathtt{LL}}$); (a) corresponds (i) in Line 4, while (b) corresponds to (ii) and (iii), to be explained

Algorithm 1 The NPFP$^{\mathsf{flex}}$ scheduling algorithm

At t_0, at which at least a job is released when the computing platform is idle, or a job finishes its execution,

1: flex $\leftarrow$ F
2: **for** Every active job (denoted by J_k of τ_k) **do**
3: **for** $C_k \in \{C_k^{\mathtt{LL}}, C_k^{\mathtt{LH}}, C_k^{\mathtt{HL}}, C_k^{\mathtt{HH}}\}$, respectively for $\{(\mathtt{D}^{\mathtt{L}},\mathtt{A}^{\mathtt{L}}),(\mathtt{D}^{\mathtt{L}},\mathtt{A}^{\mathtt{H}}),(\mathtt{D}^{\mathtt{H}},\mathtt{A}^{\mathtt{L}}),(\mathtt{D}^{\mathtt{H}},\mathtt{A}^{\mathtt{H}})\}$ **do**
4: **if** The following three conditions hold for assigned C_k: (i) Eq. (11) holds, (ii) Eq. (12) holds for all $\tau_j \in \tau$ with $Z_j(t_0) = \mathtt{T}$, and (iii) Eq. (13) holds for all $\tau_j \in \tau$ with $Z_j(t_0) = \mathtt{F}$ **then**
5: Calculate $\Delta\overline{\mathbf{\Omega}}_{\tau_k}$ in Eq. (8) for given $(\mathtt{D}^{\mathtt{L/H}},\mathtt{A}^{\mathtt{L/H}})$
6: flex $\leftarrow$ T
7: **end if**
8: **end for**
9: **end for**
10: **if** flex=T **then**
11: Execute J_k for C_k, whose $\Delta\overline{\mathbf{\Omega}}_{\tau_k}$ is the largest.
12: **else**
13: Execute the job of τ_i, whose priority is the highest among all the jobs active at t_0, during at most $C_i^{\mathtt{LL}}$ for $(\mathtt{D}^{\mathtt{L}},\mathtt{A}^{\mathtt{L}})$, which is the same as NPFP$^{\mathsf{min}}$.
14: **end if**

in the later lemmas. If it is deemed feasible to execute J_k for given C_k, we calculate $\Delta\overline{\mathbf{\Omega}}_{\tau_k}$ in Eq. (8) (Line 5), and update flex as T (Line 6). After investigating the feasibility of all pairs of an active feasible job and its execution requirement (Lines 2–9), we have two cases. If flex $=$ T, we choose to execute J_k for C_k, whose $\Delta\overline{\mathbf{\Omega}}_{\tau_k}$ is the largest among all pairs of an active feasible job and given execution requirement (Lines 10–11). Otherwise, we follow NPFP$^{\mathsf{min}}$, implying we choose the highest-priority active job according to FP and execute the job during $C_i^{\mathtt{LL}}$ for $(\mathtt{D}^{\mathtt{L}},\mathtt{A}^{\mathtt{L}})$ (Lines 12–13).

We now present online feasibility tests for NPFP$^{\mathsf{flex}}$ in Line 4, which are conditions for the job of interest J_k and other jobs than J_k not to miss their deadlines. We then prove that NPFP$^{\mathsf{flex}}$ shares the same offline schedulability analysis for NPFP$^{\mathsf{min}}$ as Lemma 1.

For ease of presentation, we define two notions. First, let $Z_i(t)$ denote the existence of an active job of τ_i at t; if $Z_i(t) = \mathtt{T}$ and $Z_i(t) = \mathtt{F}$, there exists an active job and no active job of τ_i at t, respectively. Second, let $r_i(t)$ denote the earliest release time of any job of τ_i after or at t. Since we consider the implicit-deadline periodic task model, $r_i(t)$ is not only a release time of the next job of τ_i, but also an absolute deadline of the job of τ_i that is active at t (if $Z_i(t) = \mathtt{T}$).

We focus on t_0 in Algorithm 1, at which at least one job is released when the computing platform is idle, or a job finishes its execution under NPFP$^{\mathsf{flex}}$. Suppose that we start to execute an active job of τ_k (denoted by J_k) for given C_k (either $C_k^{\mathtt{LL}}$, $C_k^{\mathtt{LH}}$, $C_k^{\mathtt{HL}}$, or $C_k^{\mathtt{HH}}$) at t_0. J_k's schedulability is simply checked in the following lemma.

Lemma 2: Suppose that we start to execute a job of τ_k (denoted by J_k) at t_0 for at most C_k. Then, if Eq. (11) holds, J_k cannot miss its deadline.

$$C_k \;\leq\; r_k(t_0) - t_0 \tag{11}$$

Proof: By the non-preemptiveness, the execution time no larger than the time to its absolute deadline implies no deadline miss. ■

Then, we check whether the earliest job of $\tau_j \in \tau$ to be executed after J_k's execution is schedulable or not, with two cases: when there exists an active job of τ_j at t_0 (i.e., $Z_j(t_0) = \texttt{T}$), and no active job of τ_j at t_0 (i.e., $Z_j(t_0) = \texttt{F}$), respectively in Lemmas 3 and 4.

Lemma 3: Suppose that (i) we start to execute an active job of τ_k (denoted by J_k) at t_0 for at most C_k, and (ii) all jobs to be executed after J_k's execution are scheduled by NPFP$^{\text{min}}$. If Eq. (12) holds, the earliest job of a given τ_j with $Z_j(t_0) = \texttt{T}$ to be executed after J_k's execution (denoted by J_j) cannot miss its deadline. Note that τ_j can be τ_k.

$$\begin{aligned} & C_j^{\text{LL}} + C_k + \sum_{\tau_h \in \mathsf{HP}(\tau_j)\setminus\{\tau_k\} | Z_h(t_0)=\texttt{T}} C_h^{\text{LL}} \\ & \quad + \sum_{\tau_h \in \mathsf{HP}(\tau_j) | r_h(t_0) < r_j(t_0)} \left\lceil \frac{r_j(t_0) - r_h(t_0)}{T_h} \right\rceil \cdot C_h^{\text{LL}} \\ \leq \;& r_j(t_0) - t_0 \end{aligned} \tag{12}$$

Proof: Suppose that J_j misses its deadline, even though Eq. (12) holds. Recall $r_j(t_0)$ is the absolute deadline of J_j that is active at t_0. Since the activeness of J_j at t_0 and (ii) in the supposition of Lemma 3, a job whose priority is lower than J_j (which is not J_k) cannot be executed in $[t_0, r_j(t_0))$ before J_j's execution. Therefore, the only jobs that can execute before J_j's execution in $[t_0, r_j(t_0))$ are (a) J_k, (b) all higher-priority jobs active at t_0 except J_k (the "except" phrase is required only if $\tau_k \in \mathsf{HP}(\tau_j)$), (c) all higher-priority jobs to be released after t_0. The WCET of (a) is C_k, and that of (b) is the first summation term of the LHS of Eq. (12). Considering $r_h(t_0)$ is the earliest job release time of given $\tau_h \in \mathsf{HP}(\tau_j)$ after t_0, the WCET of (c) is upper-bounded by the second summation term of the LHS. Therefore, missing J_j's deadline implies that the sum of the WCET of J_j itself (i.e., C_j^{LL}), the WCET of (a), the WCET of (b), and the WCET of (c) should be strictly larger than the interval length of $[t_0, r_j(t_0))$ (i.e., the RHS of Eq. (12)). The sum is upper-bounded by the LHS of Eq. (12), which contradicts Eq. (12). Therefore, J_j cannot miss its deadline if it executes for up to C_j^{LL}. ■

Lemma 4: Suppose that (i) we start to execute an active job of τ_k (denoted by J_k) at t_0 for at most C_k, (ii) all jobs to be executed after J_k's execution are scheduled by NPFP$^{\text{min}}$, and (iii) Eq. (9) holds for every $\tau_i \in \tau$. If Eq. (13) holds, the earliest job of a given τ_j with $Z_j(t_0) = \texttt{F}$ to be executed after J_k's execution (denoted by J_j) cannot miss its deadline. Note that τ_j cannot be τ_k, as the existence of J_k implies $Z_k(t_0) = \texttt{T}$.

$$\begin{aligned} & C_j^{\text{LL}} + C_k + \sum_{\tau_h \in \mathsf{HP}(\tau_j)\setminus\{\tau_k\} | Z_h(t_0)=\texttt{T}} C_h^{\text{LL}} \\ & \quad + \sum_{\tau_h \in \mathsf{HP}(\tau_j) | r_h(t_0) < r_j(t_0)+T_j} \left\lceil \frac{r_j(t_0) + T_j - r_h(t_0)}{T_h} \right\rceil \cdot C_h^{\text{LL}} \\ \leq \;& r_j(t_0) + T_j - t_0 \end{aligned} \tag{13}$$

Proof: Suppose that J_j misses its deadline, even though Eq. (13) holds. Recall $r_j(t_0) + T_j$ is the absolute deadline of J_j because J_j is not active at t_0 implying $r_j(t_0)$ is the release time of J_j. We consider two cases.

(Case 1) We consider the case where the computing platform is idle or occupied by a job that is not J_k but has lower priority than J_j. Let t' denote the latest time instant belongings to the case. By the definition of t', the interval from t' to the end of J_j's execution is included in a level-j busy period that starts from t'. By (ii) in the supposition of Lemma 4, all jobs executed after J_k's execution are scheduled by NPFP$^{\text{min}}$. On the other hand, the proof of Lemma 1 shows that if Eq. (9) holds for $\tau_j \in \tau$, any job of τ_j in any level-j busy period does not miss its deadline when τ is scheduled by NPFP$^{\text{min}}$. Therefore, if we consider the proof of Lemma 1, J_j's deadline miss contradicts (iii) in the supposition of Lemma 4.

(Case 2: the opposite of Case 1) In this case, the proof is the same as that of Lemma 3 by changing the interval of interest from $[t_0, r_j(t_0))$ to $[t_0, r_j(t_0) + T_j)$.

By Cases 1 and 2, J_j cannot miss its deadline if it executes for up to C_j^{LL}. ■

While Lemmas 3 and 4 guarantee the schedulability of the earliest job of each τ_j to be executed after J_k's execution, we need to guarantee the schedulability of every job of τ_j to be executed after J_k's execution, as follows.

Lemma 5: Suppose that (i) a job of τ_j (denoted by J_j) starts its execution at t_1 and does not miss its deadline if it executes for up to C_j^{LL}, (ii) all jobs to be executed after J_j's execution are scheduled by NPFP$^{\text{min}}$, and (iii) Eq. (9) holds for every $\tau_i \in \tau$. Then, any job of τ_j to be executed after J_j's execution cannot miss its deadline.

Proof: Let J_j' denote the next job of J_j invoked by τ_j. We focus on $[t_1, t_2)$, where t_2 is the time instant at which J_j' starts its execution. We prove no deadline miss of J_j'.

(Case 1: no processor idle status and no execution of jobs whose priority is lower than τ_j in $[t_1, t_2)$) Since J_j and J_j' are executed in the same level-j busy period, we can prove that $t_2 - t_1 \leq T_j$, by applying the technique in the second part of the proof of Lemma 1 (corresponding to $t_{x+1} - t_x \leq T_i$). Therefore, (i) in the supposition of Lemma 5 implies no deadline miss of J_j' if it executes up to C_j^{LL} (which is guaranteed by (ii) in the supposition of Lemma 5).

(Case 2: the opposite case of Case 1) Let t' denote the latest time instant in $[t_1, t_2)$ at which the computing platform is idle or occupied by a job whose priority is lower than J_j'. The remaining proof is the same as Case 1 of the proof of Lemma 4.

By Cases 1 and 2, J'_j cannot miss its deadline. Applying the same technique as proving no deadline miss of J'_j from no deadline miss of J_j, we can prove that all following jobs of τ_j do not miss their deadlines. ■

Utilizing Lemmas 2, 3, 4, and 5, we develop an offline schedulability analysis for NPFP$^{\text{flex}}$, which is the same as that for NPFP$^{\text{min}}$ (i.e., Lemma 1).

Theorem 1: Suppose that a task set τ is scheduled by NPFP$^{\text{flex}}$ in Algorithm 1. If every task $\tau_i \in \tau$ satisfies Eq. (9), every job invoked by tasks in τ cannot miss its deadline.

Proof: Suppose that a job misses its deadline at t_a. Then, there should exists the latest time instant t_0 before t_a, at which a job of τ_k (denoted by J_k) starts its execution although it is not the highest-priority active job and/or it executes for more than C_k^{LL}. Otherwise, until t_a, all jobs are scheduled according to NPFP$^{\text{min}}$; the existence of a job deadline miss contradicts Lemma 1.

By the definition of t_0, in the time interval from the end of J_k's execution to t_a, all jobs are scheduled according to NPFP$^{\text{min}}$. Also, by the policy of NPFP$^{\text{flex}}$, J_k starts its execution at t_0 only if the following three conditions hold for assigned C_k (in Line 4 of Algorithm 1): (i) Eq. (11) holds, (ii) Eq. (12) holds for all $\tau_j \in \tau$ with $Z_j(t_0) = \texttt{T}$, and (iii) Eq. (13) holds for all $\tau_j \in \tau$ with $Z_j(t_0) = \texttt{F}$. (i) implies J_k cannot miss its deadline by Lemma 2, (ii) and (iii) imply the earliest job of every τ_j to be executed after J_k's execution cannot miss its deadline by Lemmas 3 and 4. By Lemma 5, (ii) and (iii) imply every job of τ_j to be executed after J_k's execution cannot miss its deadline. Therefore, the existence of a job deadline miss at t_a contradicts either Lemma 2, 3, 4, or 5, which proves the theorem. ■

Run-time complexity of NPFP$^{\text{flex}}$. At each t_0 (at which a job is released or completed in Algorithm 1), we need to perform Line 4 of Algorithm 1 for every active job J_k. Since checking Eq. (12) or (13) takes $O(n)$ for given J_k and τ_j, it takes $O(n^2)$ to perform Line 4 of Algorithm 1 for given J_k, where n is the number of tasks in τ. Therefore, at each t_0, NPFP$^{\text{flex}}$ requires $O(n^2 \cdot n')$, where n' is the number of active jobs at t_0. Note that the number of cameras of an autonomous vehicle (i.e., n) is only a few, e.g., one each in the front, rear and both sides of the vehicle, and one each between the front and both sides; therefore, RT-MOT provides an acceptable scheduling overhead with $O(n^2 \cdot n')$ time-complexity, to be demonstrated with experimental results in Sec. VI-A.

VI. Evaluation

In this section, we first explain our experimental setup. We then present experimental results of RT-MOT, compared to existing methods.

A. Experimental Setup

Hardware and software. We conduct experiments on a computer equipped with an Intel(R) Xeon(R) Silver 4215R CPU @ 3.20GHz, 251.5GB RAM, and NVIDIA V100 GPU. We use Ubuntu 18.04.4 with CUDA 10.2, and PyTorch 1.10.2 for implementation. We employ YOLOv5 as a front-end detector with the different input sizes of 672× 672 and 256× 256, for high- and low-confidence detection, respectively. We also employ SORT [3] and DeepSORT [2] with the re-identification model of OSNet [15] for high- and low-confidence association, respectively. We use the Waymo Open Dataset [16] in our evaluation.

TABLE I
Execution time measurement for each component

Time(ms)	C_i^D			C_i^A		
	c^{pre}	$c_i^{infer}(\texttt{L})$	$c_i^{infer}(\texttt{H})$	$c_i^{as}(\texttt{L})$	$c_i^{as}(\texttt{H})$	c^{post}
Average	0.6	12.6	13.1	3.2	23.4	0.7
Maximum	0.9	17.6	23.2	9.6	32.7	0.9

Execution time profiling and run-time overhead. RT-MOT conducts DNN-based computations on a GPU, while others including IoU matching and confidence estimation use a CPU. We take a measurement-based approach to estimate the worst-case execution time of detection and association parts for each multi-object tracking task, C_i^D and C_i^A, under RT-MOT. Our experiments measure the execution time of each component in C_i^D and C_i^A by running it 1,000 times, taking the maximum value among the measured ones as the WCET, summarized in Table I. For example, the WCETs for low- and high-confidence detection ($c_i^{infer}(\texttt{L})$ and $c_i^{infer}(\texttt{H})$) are 17.6ms and 23.2ms, while the WCETs for low- and high-confidence association ($c_i^{as}(\texttt{L})$ and $c_i^{as}(\texttt{H})$) are 9.6ms and 32.7ms, respectively. The two key modules for RoI extraction and confidence estimation (c^{pre} and c^{post}) in RT-MOT take 0.6ms and 0.7ms on average, respectively, a relatively short delay compared to the execution times for detection and association (c_i^{infer} and c_i^{as}).

We also measure the run-time overhead of our scheduling framework for RT-MOT as a function of the number of MOT tasks (n). When n is increased from 1 to 10, the run-time scheduling overhead increases from 0.5ms to 0.7ms, which is almost linear in n with low slope.

B. Experimental Results

We would like to demonstrate the capability of RT-MOT making a significant improvement in multi-object tracking accuracy while meeting all timing requirements for multiple MOT tasks. We use MOTA [17], a primary metric to evaluate the accuracy of multi-object tracking algorithms. The MOTA metric deals with both detection and tracker outputs by taking into consideration of miss detection, false detection, and false tracking. Note that a higher MOTA score indicates higher overall tracking accuracy.

We evaluate three versions of RT-MOT, which (i) employ different job prioritization and execution requirements for detection and association, but (ii) share the same offline schedulability analysis of Eq. (9) in Lemma 1.

- RT-MOT$^{\text{min}}$ employs NPFP$^{\text{min}}$ (i.e., always performing L detection and L association).
- RT-MOT$^{\text{flex-NPI}}$ employs NPFP$^{\text{flex}}$ but does not allow priority inversion, i.e., the feasibility of J_k for H detection

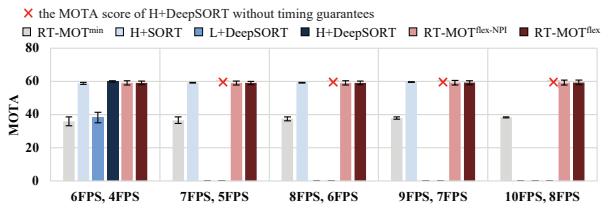

Fig. 5. Tracking accuracy comparison for task sets with two tasks

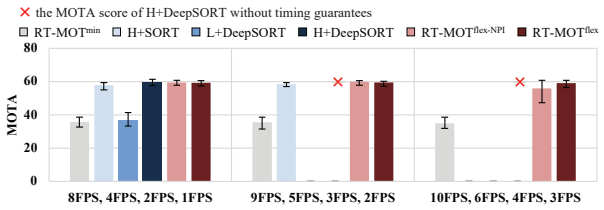

Fig. 6. Tracking accuracy comparison for task sets with four tasks

and/or H association in Lines 3–8 of Algorithm 1 is tested only for the job whose priority is the highest among all active jobs rather than every active job in Line 2.

- RT-MOT$^{\text{flex}}$: RT-MOT employs NPFP$^{\text{flex}}$ in Algorithm 1 as it is.

As a prioritization policy for FP, we employ RM (Rate Monotonic).

We compare the RT-MOT versions with two existing popular multi-object tracking approaches.

- H+SORT employs YOLOv5 with the original frame size (672×672) for detection and SORT for association (that corresponds to L association of RT-MOT).
- L+DeepSORT employs YOLOv5 with the down-scaled frame size (256×256) for detection and DeepSORT for association (that corresponds to H association of RT-MOT).
- H+DeepSORT employs YOLOv5 with the original frame size (672×672) for detection and DeepSORT for association.

Note that we exclude L+SORT for comparison because it shows a similar result to RT-MOT$^{\text{min}}$. For a fair comparison, H+SORT, L+DeepSORT, and H+DeepSORT employ non-preemptive fixed-priority scheduling with the same offline schedulability analysis of Eq. (9).

Fig. 5 compares the average MOTA scores (plotted as bar graphs) and the maximum and minimum MOTA scores (plotted as error bars) under six different approaches for five different task sets. Each of the task sets consists of two tasks with different FPS requirements, from the smallest workload (6FPS and 4FPS) in the left-most, to the largest one (10FPS and 8FPS) in the right-most, increased by 1FPS each for both tasks. Note that, if a task set is not deemed schedulable by the offline schedulability analysis in Eq. (9) under the target MOT approach, we do not include to plot MOTA scores as we cannot offer an offline timing guarantee. We observe that RT-MOT$^{\text{flex-NPI}}$ and RT-MOT$^{\text{flex}}$ consistently show much higher overall accuracy than RT-MOT$^{\text{min}}$ across all task sets (1.5× accuracy improvement). H+DeepSORT shows the highest overall accuracy (with a marginal improvement over RT-MOT$^{\text{flex}}$ by 1.0%p) for the task set with 6FPS and 4FPS, but it cannot achieve timely execution for other task sets due to heavy computational workloads. H+SORT also shows a comparable accuracy with RT-MOT$^{\text{flex}}$ for three task sets with up to 9FPS and 7FPS, but it cannot achieve timely execution for the task set with 10FPS and 8FPS.

Fig. 6 shows the accuracy results for three task sets, each of which consists of four tasks. In general, we observe a similar trend to Fig. 5; all MOT approaches except RT-MOT$^{\text{min}}$ and L+DeepSORT exhibit comparable MOTA scores under the smallest workload (8FPS, 4FPS, 2FPS, and 1FPS), while only RT-MOT$^{\text{flex-NPI}}$ and RT-MOT$^{\text{flex}}$ perform well under the largest workload (10FPS, 6FPS, 4FPS, and 3FPS). However, for the task set with the largest workload, RT-MOT$^{\text{flex}}$ exhibits a smaller difference in accuracy among tasks by only up to 4.4%p, while RT-MOT$^{\text{flex-NPI}}$ exhibits a larger difference by up to 13%p; also, we observe 3.1%p accuracy degradation of RT-MOT$^{\text{flex-NPI}}$, compared to RT-MOT$^{\text{flex}}$. Such a result can be interpreted as the benefit of RT-MOT$^{\text{flex}}$ that enables flexible job-level scheduling by allowing bounded priority inversions without violating any FPS requirement. Although RT-MOT$^{\text{min}}$, H+SORT, L+DeepSORT, and H+DeepSORT are *static* approaches that apply a fixed choice of detection and association models across all frames, RT-MOT$^{\text{flex-NPI}}$ and RT-MOT$^{\text{flex}}$ dynamically determine a pair of detection and association models frame-by-frame by considering both available resources and expected confidence improvement at run-time, achieving higher overall tracking accuracy while satisfying all FPS requirements. For example, for the task set with the largest workload, RT-MOT$^{\text{flex}}$ flexibly selects ($\mathtt{D}^{\mathtt{L}}$,$\mathtt{A}^{\mathtt{L}}$), ($\mathtt{D}^{\mathtt{H}}$,$\mathtt{A}^{\mathtt{L}}$), and ($\mathtt{D}^{\mathtt{H}}$,$\mathtt{A}^{\mathtt{H}}$) for 52, 70, and 336 frames, respectively, out of 458 total jobs. Note that ($\mathtt{D}^{\mathtt{L}}$,$\mathtt{A}^{\mathtt{H}}$) is rarely selected in our experiment because i) $c_i^{as}(\mathtt{H})$ is larger than $c_i^{infer}(\mathtt{H})$ by 1.4× as observed in Table I, and ii) L+DeepSORT shows a marginal accuracy improvement over RT-MOT$^{\text{min}}$ as observed in Figs. 5 and 6. Also, note that the maximum achievable MOTA score by H+DeepSORT *without timing guarantees* is 60 (marked as red cross) for the task set with the largest workload, while the MOTA score of RT-MOT$^{\text{flex}}$ is 59.1. Therefore, RT-MOT$^{\text{flex}}$ can achieve nearly maximum tracking accuracy with 0.84× less total computation time as compared to H+DeepSORT, making the task set schedulable.

In summary, RT-MOT$^{\text{flex}}$ can be adapted to various task sets through the dynamic selection of a pair of detection and association execution models and flexible scheduling frame-by-frame, achieving high overall tracking accuracy while meeting all FPS requirements at run-time in addition to offline timing guarantees.

VII. Related Work

MOT can be categorized into one-stage and two-stage models depending on where in detection and association the AI (Artificial Intelligence) model is used. RT-MOT uses two-stage MOT algorithms such as DeepSORT [2] and StrongSORT [18], which employ two different models each for both detection and association. Two-stage MOT uses the detection model to extract the detected object with the re-identification model and then associates it with matching algorithms. On the other hand, a one-stage MOT only uses a single model during detection and association. Objects and their features can be found within a single inference, and much like a two-stage MOT, the association process utilizes matching algorithms, e.g., FairMOT [19] and BytesTrack [20].

While most one-stage and two-stage models for MOT have focused on high tracking accuracy and average FPS, several studies have attempted to address both timing guarantee and high detection accuracy for the object detection [21]–[24], which is one of the main parts of two-stage MOT. Since those studies do not take the execution for association into consideration, their techniques for timing guarantee cannot be directly applied to MOT. When it comes to MOT itself, there has been a sole study that has achieved both timing guarantee and high tracking accuracy [1]. By defining and utilizing the notion of uncertainty, the study in [1] has achieved a timing guarantee of the scheduling horizon while improving location accuracy. However, since the study has targeted a single MOT task, it cannot be applied to multiple MOT tasks, which is different from RT-MOT.

VIII. Conclusion

In this paper, we proposed a novel confidence-aware real-time scheduling framework for multiple MOT tasks, RT-MOT, which consists of (i) a method to estimate the overall accuracy variation according to different detector/tracker selections, (ii) a scheduling framework that provides offline timing guarantees while maximizing overall accuracy at run-time using the method, and (iii) a system architecture that supports the framework. Through experiments, we demonstrated that RT-MOT achieves high accuracy and timely execution of a set of MOT tasks, which has not been accomplished by existing tracking-by-detection methods. In the future, we would like to extend our framework towards more combinations of various detectors and trackers. In addition, we would like to improve the scheduling framework in terms of schedulability performance, e.g., by allowing a preemption between the completion of detection and the beginning of association for each MOT task.

Acknowledgement

This work was supported by the National Research Foundation of Korea (NRF) grant funded by the Korea government (MSIT) (2018R1A5A1060031 (ERC), NRF-2020R1F1A1071547, NRF-2020R1F1A1076058, NRF-2021R1A2B5B02001758, NRF-2021R1F1A1059277, NRF-2021K2A9A1A01101570, NRF-2022R1A4A3018824, NRF-2022R1C1C1009208).

This work was also supported by Institute of Information & communications Technology Planning & Evaluation (IITP) grant (IITP-2022-0-01053, IITP-2022-0-00448, Deep Total Recall: Continual Learning for Human-Like Recall of Artificial Neural Networks, 10%) funded by the Korea government (MSIT), as well as the DGIST R&D Program of MSIT (20-CoE-IT-01).

References

[1] S. Liu, X. Fu, M. Wigness, P. David, S. Yao, L. Sha, and T. Abdelzaher, "Self-cueing real-time attention scheduling in criticality-aware visual machine perception," in *IEEE Real Time Technology and Applications Symposium (RTAS)*, 2022, pp. 173–186.

[2] N. Wojke, A. Bewley, and D. Paulus, "Simple online and realtime tracking with a deep association metric," in *IEEE international conference on image processing (ICIP)*. IEEE, 2017, pp. 3645–3649.

[3] A. Bewley, Z. Ge, L. Ott, F. Ramos, and B. Upcroft, "Simple online and realtime tracking," in *IEEE international conference on image processing (ICIP)*. IEEE, 2016, pp. 3464–3468.

[4] W. Kang, S. Chung, J. Y. Kim, Y. Lee, K. Lee, J. Lee, K. G. Shin, and H. S. Chwa, "DNN-SAM: Split-and-merge dnn execution for real-time object detection," in *IEEE Real Time Technology and Applications Symposium (RTAS)*, 2022, pp. 160–172.

[5] P. Voigtlaender, M. Krause, A. Osep, J. Luiten, B. B. G. Sekar, A. Geiger, and B. Leibe, "Mots: Multi-object tracking and segmentation," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2019, pp. 7942–7951.

[6] Z. Wang, L. Zheng, Y. Liu, Y. Li, and S. Wang, "Towards real-time multi-object tracking," in *European Conference on Computer Vision (ECCV)*. Springer, 2020, pp. 107–122.

[7] Z. Lu, V. Rathod, R. Votel, and J. Huang, "Retinatrack: Online single stage joint detection and tracking," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2020, pp. 14 668–14 678.

[8] G. Brasó and L. Leal-Taixé, "Learning a neural solver for multiple object tracking," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2020, pp. 6247–6257.

[9] S.-H. Bae and K.-J. Yoon, "Confidence-based data association and discriminative deep appearance learning for robust online multi-object tracking," *IEEE Transactions on Pattern Analysis and Machine Intelligence (TPAMI)*, vol. 40, no. 3, pp. 595–610, 2017.

[10] C. Liu and J. Layland, "Scheduling algorithms for multi-programming in a hard-real-time environment," *Journal of the ACM*, vol. 20, no. 1, pp. 46–61, 1973.

[11] A. Burns, K. Tindell, and A. Wellings, "Effective analysis for engineering real-time fixed priority schedulers," *IEEE Transactions on Software Engineering (TSE)*, vol. 21, no. 5, pp. 475–480, 1995.

[12] G. Yao, G. Buttazzo, and M. Bertogna, "Feasibility analysis under fixed priority scheduling with fixed preemption points," in *Proceedings of IEEE International Conference on Embedded and Real-Time Computing Systems and Applications (RTCSA)*, 2010, pp. 71–80.

[13] J.-J. Chen and G. von der Bruggen, "Non-preemptive and limited preemptive scheduling," *Lecture Note in TU Dortmund*, 2017. [Online]. Available: https://ls12-www.cs.tu-dortmund.de/daes/media/documents/teaching/courses/rts/non-preemptive-scheduling.pdf

[14] L. George, N. Rivierre, and M. Spuri, "Preemptive and non-preemptive real-time uniprocessor scheduling," Inria, Tech. Rep., 1996.

[15] K. Zhou, Y. Yang, A. Cavallaro, and T. Xiang, "Omni-scale feature learning for person re-identification," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2019, pp. 3702–3712.

[16] P. Sun, H. Kretzschmar, X. Dotiwalla, A. Chouard, V. Patnaik, P. Tsui, J. Guo, Y. Zhou, Y. Chai, B. Caine *et al.*, "Scalability in perception for autonomous driving: Waymo open dataset," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2020, pp. 2446–2454.

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

[18] Y. Du, Y. Song, B. Yang, and Y. Zhao, "Strongsort: Make deepsort great again," *arXiv preprint arXiv:2202.13514*, 2022.
[19] 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 (IJCV)*, vol. 129, no. 11, 2021.
[20] Y. Zhang, P. Sun, Y. Jiang, D. Yu, Z. Yuan, P. Luo, W. Liu, and X. Wang, "Bytetrack: Multi-object tracking by associating every detection box," *arXiv preprint arXiv:2110.06864*, 2021.
[21] E. Kim, C. Ahn, and S. Oh, "NestedNet: Learning nested sparse structures in deep neural networks," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2018.
[22] J.-E. Kim, R. Bradford, and Z. Shao, "AnytimeNet: controlling time-quality tradeoffs in deep neural network architectures," in *Design, Automation, and Test in Europe (DATE)*, 2020.
[23] S. Lee and S. Nirjon, "SubFlow: A dynamic induced-subgraph strategy toward real-time DNN inference and training," in *IEEE Real Time Technology and Applications Symposium (RTAS)*, 2020.
[24] S. Heo, S. Cho, Y. Kim, and H. Kim, "Real-time object detection system with multi-path neural networks," in *IEEE Real Time Technology and Applications Symposium (RTAS)*, 2020.