Journal of Systems Architecture 160 (2025) 103349

Contents lists available at ScienceDirect

Journal of Systems Architecture

journal homepage: www.elsevier.com/locate/sysarc

Real-time scheduling for multi-object tracking tasks in regions with different criticalities

Donghwa Kang [a], Jinkyu Lee [b,*], Hyeongboo Baek [c,*]

[a] Korea Advanced Institute of Science and Technology (KAIST), Daejeon, South Korea
[b] Sungkyunkwan University (SKKU), Suwon, South Korea
[c] University of Seoul (UOS), Seoul, South Korea

ARTICLE INFO

Keywords:
Multi-object tracking
Real-time scheduling
Timing guarantee
Criticality-awareness
Autonomous driving

ABSTRACT

Autonomous vehicles (AVs) utilize sensors such as LiDAR and cameras to iteratively perform sensing, decision-making, and actions. Multi-object tracking (MOT) systems are employed in the sensing stage of AVs, using these sensors to detect and track objects like pedestrians and vehicles, thereby enhancing situational awareness. These systems must handle regions of varying criticality and dynamically shifting locations, all within limited computing resources. Previous DNN-based MOT approaches primarily focused on tracking accuracy, but timing guarantees are becoming increasingly vital for autonomous driving. Although recent studies have introduced MOT scheduling frameworks with timing guarantees, they are either restricted to single-camera systems or fail to prioritize safety-critical regions in the input images. We propose CA-MOT, a Criticality-Aware MOT execution and scheduling framework for multiple cameras. CA-MOT provides a control knob that balances tracking accuracy in safety-critical regions and timing guarantees. By effectively utilizing this control knob, CA-MOT achieves both high accuracy and timing guarantees. We evaluated CA-MOT's performance using a GPU-enabled embedded board commonly employed in AVs, with data from real-world autonomous driving scenarios.

1. Introduction

Autonomous vehicles (AVs) are systems that iteratively perform sensing, decision-making, and actions using various sensors such as LiDAR, radar, inertial measurement units (IMU), and cameras [1]. Multi-object tracking (MOT) systems, used in the perception stage of AVs, track objects like pedestrians and cars, enhancing situational awareness. Since MOT information is periodically transferred to control tasks, timely execution must be guaranteed to ensure safety and prevent severe accidents [2–4]. Low accuracy, despite timely execution, may result in missed objects, thus compromising AVs' safety [2,4,5]. Therefore, AV MOT systems should ensure timing guarantees with maximized accuracy.

Tracking-by-detection [6,7] is widely used due to its high accuracy and ability to leverage state-of-the-art DNN-based detection models (e.g., YOLO series [8], Faster R-CNN [9]). For each input image from each camera, tracking-by-detection performs two tasks: detection and association. Detection uses DNN-based models to sense the *motion* information of objects, such as location and velocity, while association matches objects between frames based on extracted *feature* information (also called feature vectors or feature maps obtained through pooling and convolutional layers) using CNN (convolutional neural network)-based models (e.g., OS-Net [10]). For unmatched objects, location-based methods like intersection over union (IoU) are applied.

MOT input images exhibit two key characteristics: (i) regions with varying levels of criticality and (ii) dynamically shifting locations. With limited computing resources in AVs, it is crucial to deliver different levels of service quality based on criticality. Safety-critical regions, where objects with a short time-to-collision (e.g., under 2 s) cluster, must be prioritized. If multiple clusters exist, the broader area encompassing them is considered the safety-critical region, as defined in DNN-SAM [5]. Established methods compute time-to-collision using LiDAR and IMU data; we follow the approach from DNN-SAM. This leads to two requirements for criticality-aware MOT systems: (R1) accuracy maximization for safety-critical regions and (R2) timing guarantees.

Most existing DNN-based MOT approaches focus on accuracy [7,11,12], but timing guarantees are increasingly critical in autonomous driving. Recent research has proposed MOT resource scheduling frameworks that guarantee timing for every MOT execution [2,4]. However, [2] overlooks safety-criticality, while [4] focuses on a single task. We address safety-criticality across multiple tasks, raising the following

* Corresponding authors.
E-mail addresses: anima0729@kaist.ac.kr (D. Kang), jinkyu.lee@skku.edu (J. Lee), hbbaek359@gmail.com (H. Baek).

https://doi.org/10.1016/j.sysarc.2025.103349
Received 22 September 2024; Received in revised form 13 December 2024; Accepted 20 January 2025
Available online 28 January 2025

challenges:

C1. How to balance R1 and R2 to efficiently use limited computing resources.
C2. How to achieve both R1 and R2 by effectively using the control knob developed from C1.

In this paper, we propose CA-MOT, a Criticality-Aware MOT execution and scheduling framework for multiple MOT tasks. To address C1, CA-MOT offers three execution options (low, middle, and high workloads) to balance R1 and R2 for both detection and association. To address C2, CA-MOT introduces the notion of *aging* for detection and association sub-tasks, estimating the reliability of motion and feature information over time. Balancing the aging of these tasks is essential to achieve R1 and R2 with limited resources (to be discussed in Section 3.4). Based on this, CA-MOT develops two scheduling algorithms: EDF-BE and EDF-Slack. EDF-BE increases the workload of tasks waiting in the ready queue for execution (referred to as active tasks) without compromising the R2 bound when no other tasks are pending. In contrast, EDF-Slack is designed to handle scenarios with multiple active tasks.

To validate CA-MOT's performance in meeting R1 and R2, we conducted extensive experiments on an NVIDIA Jetson Xavier using the KITTI Dataset [13]. Additionally, we applied three detectors in our experiments: YOLOv5 [14], YOLOX [8], and Faster-RCNN [9].

The contributions of this paper are as follows:

- We motivate the importance of balancing between aging of detection and association to achieve R1 and R2 (Section 2).
- We propose a new system design, CA-MOT that addresses R1 and R2 considering varying levels of criticality in different regions for multiple MOT tasks (Section 3).
- We develop new scheduling algorithms to effectively achieve R1 and R2 by balancing between aging of detection and association for each MOT task (Section 4).
- We demonstrate the effectiveness of CA-MOT in achieving R1 and R2 using a real-world self-driving dataset (Section 5).

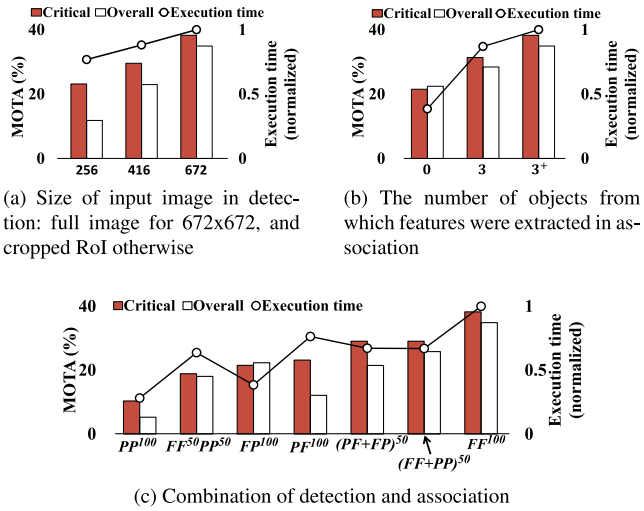
2. Motivation

This section presents target systems and motivates the system design of CA-MOT to address C1 and C2 based on measurement-based observations.

2.1. Target system

CA-MOT targets 2D MOT systems on AVs equipped with multiple camera sensors. Each MOT task performs MOT execution on consecutive input frames received from the corresponding camera sensor at a predetermined period. As this recurring task is required to complete a job within a specified deadline, each MOT task is considered a real-time task with a period and deadline. CA-MOT employs tracking-by-detection comprising two steps of MOT execution: detection and association. The front-end detector performs detection by exploiting the existing stand-alone DNN-based detector to identify the position and class of objects in the input image. Using the locations of detected objects, the feature extractor (e.g., the deployed CNN model such as OSNet) extracts features (i.e., feature vectors or feature maps) for each object. These features capture the visual characteristics of each object. The back-end tracker compares the feature similarities between objects in the current frame and the previous frame, matching objects with high similarity. For any remaining unmatched objects, a location-based matching method such as IoU is applied. The tracker then stores the motion information (position and velocity) and the features of each object in preparation for the next frame.

We assume a system in which each camera independently tracks objects moving within its field of view. While it is possible to consider a system that tracks specific objects moving between the fields of view of multiple cameras (called hand-over), this is beyond the scope of our work. This paper focuses on dividing the multi-object tracking task into two subtasks (i.e., detection and association) and using DNN-based MOT-specific properties (i.e., reuse of motion and feature information) to achieve R1 and R2 under limited resources.



(a) Size of input image in detection: full image for 672x672, and cropped RoI otherwise

(b) The number of objects from which features were extracted in association

(c) Combination of detection and association

Fig. 1. Tracking accuracy and execution time on different execution options of detection and association.

2.2. Trade-off between accuracy and execution time

To address C1, we consider two factors: (i) the input image size and detection within the safety-critical region, and (ii) the number of objects used for feature extraction during association across all detected objects in each frame.

Fig. 1(a) compares the multi-object tracking accuracy (MOTA) [15] for the overall and safety-critical regions (referred to as overall and critical accuracy) and the execution time for a single MOT task using three input image sizes ($256 \times 256, 416 \times 416, 672 \times 672$). Overall accuracy considers all objects, while critical accuracy focuses on the safety-critical region. YOLOv5 [14] is used for detection, and features are extracted for all detected objects. The KITTI dataset [13] is used. For image sizes 256×256 and 416×416, detection is performed on a cropped region of interest (RoI) that includes the safety-critical region. If the RoI is smaller, it is resized to include the critical region; otherwise, the critical region is cropped accordingly. The safety-critical region will be defined in Section 3. For the 672×672 size (i.e., the original input size), detection occurs without cropping.

As shown in Fig. 1(a), reducing image size leads to a notable decrease in overall accuracy, while critical accuracy decreases less significantly due to prioritization of the safety-critical region in the RoI. Additionally, execution time decreases as the image size is reduced, demonstrating a trade-off between R1 and R2 when focusing on the critical region.

Fig. 1(b) shows the impact of varying the number of objects used for feature extraction on accuracy and execution time, with the image size fixed at 672×672. The number of objects ranges from zero, three, and more than three. OS-Net [10] is used for feature extraction. As shown in Fig. 1(b), as the number of objects with feature extraction increases, both overall and critical accuracy improve, but this also leads to increased execution time. This highlights a trade-off between R1 and R2 based on the number of objects considered for feature extraction.

Section 3.3 details the MOT execution pipeline of CA-MOT, which leverages these observations to effectively address C1.

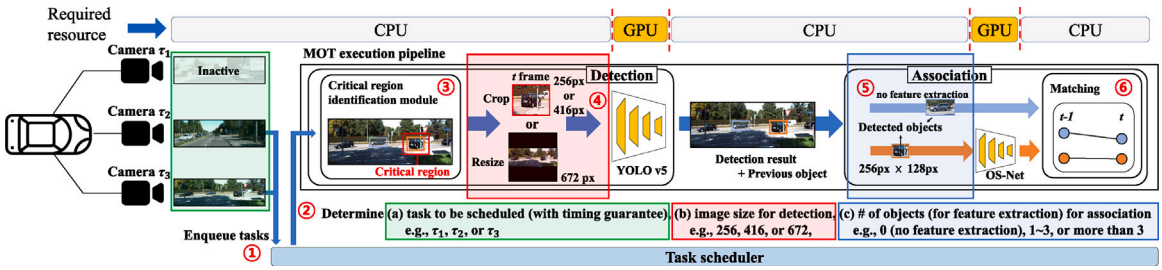


Fig. 2. System design of CA-MOT: the key features are (a) an aging-aware scheduler that provides timing guarantees and a criticality-aware flexible MOT execution pipeline including (b) a detection module that accommodates varying input sizes and (c) an association module that handles a varying number of objects for feature extraction.

2.3. Different combination of detection and association

To address C2, Fig. 1(c) reveals an intriguing observation that different combinations of image sizes and the number of feature extractions yield distinct effects on accuracy and execution time. The experiment was conducted over 100 consecutive frames.

In Fig. 1(c), the execution of detection or association is denoted by P or F. P represents partial computation, where detection is performed only on the region of interest (RoI) at a size of 256 × 256, including a safety-critical region, and association is limited to location-based association without feature extraction. F represents full computation, where detection is performed on the entire image at a size of 672 × 672, and association includes feature extraction for all objects. The number in the upper right of the notation indicates how many times the combination of detection and association has been performed. For example, the notation $FF^{50}PP^{50}$ indicates that we use F for both the detection and association steps in the first 50 frames and P for both phases in the remaining 50 frames. To mitigate the issue of objects outside the critical region not being detected due to cropping, which can decrease the accuracy of the non-critical region, we utilize the position information of objects from the previous frame as predicted position information for the current frame using a prediction model such as Kalman filter [16] during the execution of P in the detection step. Except for PP^{100} and FF^{100}, all combinations have the same proportion of F and P for the entire frames.

As shown in Fig. 1(c), although $FF^{50}PP^{50}$ and PF^{100} have similar execution times with $(PF+FP)^{100}$ and $(FF+PP)^{100}$, they show lower tracking accuracy. This indicates that different combinations of F and P can have a varying impact on accuracy. The observation in Fig. 1(c) necessitates a new scheduler that is capable of obtaining high tracking accuracy by capturing an MOT-specific property referred to as *aging*, which will be detailed in Section 3.4.

3. System design of CA-MOT

This section presents the goal and design of CA-MOT to address C1 and C2.

3.1. System overview

CA-MOT utilizes a tracking-by-detection approach, consisting of two steps: detection and association, where the front-end detector employs a pre-existing DNN-based detector to detect and classify objects in the input image, and the unmatched objects are matched using a location-based method like IoU. CA-MOT aims at providing prioritized tracking accuracy for the safety-critical region with a timing guarantee for every MOT execution on limited computing resources by addressing C1 and C2 discussed in Section 1, which has the following design goals.

- It provides different execution options not only for detection but also association considering different criticality of regions in input images.
- It provides a timing guarantee while providing prioritized tracking accuracy for the safety-critical regions by exploiting an MOT-specific property.

To address the first goal, CA-MOT implements a criticality-aware flexible MOT execution pipeline in which detection and association are performed with different execution option by leveraging the observations discussed in Section 2.2. To address the second goal, CA-MOT develops an aging-aware task-level scheduler to provide accuracy maximization while providing a timing guarantee by exploiting the observations discussed in Section 2.3 building upon the MOT execution pipeline. The MOT execution pipeline and the scheduler are implemented as separate threads, and they are communicated with shared memory.

CA-MOT does not require modifications to existing DNN models (e.g., detectors and feature extractors), which allows for reusing most (if not all) stand-alone detectors and feature extractors. Notably, state-of-the-art detectors like YOLOv5 are inherently designed to handle varying input image sizes, and all CNN models can perform batch execution on multiple images (each corresponding to a different object). As shown in Fig. 2, the key features of CA-MOT are: (a) a scheduling policy that selects one input per camera, provides timing guarantees, and adjusts the workload for detection and association; (b) a module that processes detection with inputs of varying sizes; and (c) a module that extracts features from a pre-determined number of detected objects.

3.2. Workflow

Fig. 2 presents the workflow of CA-MOT. During system operation, the task scheduler maintains a queue to store images periodically received from each camera sensor (①). Then, the task scheduler determines the following for tasks in the queue: (a) the task to be scheduled, (b) the execution option for the detector, and (c) the execution option for the association (②). After an image moves to the MOT execution pipeline, the critical region identification module identifies a safety-critical region from the image and crops (or not) a RoI including the safety-critical region according to the execution option for the detector (③). Depending on the execution option, the cropped RoI or entire image is processed for detection (④). Furthermore, depending on the number of objects for which features are extracted, CA-MOT selectively extracts features for detected objects (⑤). All detected objects are then matched with the tracked objects from the previous frame. If both the detected and tracked objects have feature vectors, they are associated through feature-based matching. Otherwise, they are associated solely based on their locations (⑥).

3.3. Criticality-aware flexible MOT execution pipeline

The MOT execution pipeline conducts detection and association sequentially. CA-MOT can employ any existing stand-alone DNN-based detectors as long as it can accommodate different sizes of input images (e.g., YOLO series) and offer a clear trade-off between accuracy and execution time. For each input image with a size of 672 × 672, the

detector performs the detection to identify the location and class of multiple objects in the image. Once the scheduler determines the task (associated with an input image) to be scheduled and the execution option for the tasks, the detection is performed for the task according to the execution option. CA-MOT provides three execution options (i.e., low, middle, and high workloads, respectively) providing a trade-off between execution time and accuracy. For low and middle workload detections, CA-MOT first identifies the RoI with sizes of 256×256 and 416×416, respectively, and then detection is performed on cropped RoI, which includes the safety-critical region. The area outside the RoI is not subject to detection, and the motion information (e.g., size, position, velocity, direction) of objects detected in the previous frame is used in the prediction models such as the Kalman filter to obtain the estimated information of objects in the current frame. On the other hand, high-workload detection is performed on the original image with a size of 672×672.

We define the area that encompasses all safety-critical objects, which are objects with a time-to-collision of less than two seconds, as the safety-critical area. If the safety-critical area exceeds the input size for the detector, as determined by the detection process (e.g., 256×256 or 416×416), the safety-critical area is cropped and resized to the corresponding dimensions before being fed into the detector model. The locations of safety-critical objects are determined based on their most recently computed positions, without projecting future safety-critical regions from them. There are numerous existing approaches that calculate time-to-collision based on the relative positions of objects and the ego vehicle given LiDAR and IMU data, and we assume the use of one such method. It is also important to note that the KITTI dataset provides both LiDAR and IMU data. For example, areas where objects with a time-to-collision of less than 2 s congregate can be defined as *safety-critical regions*, and if multiple such areas exist, the encompassing area that includes all of them would be considered the safety-critical region. Please note that we adhere to the definition of the safety-critical region as defined in the existing paper DNN-SAM in [5]. It is assumed that the critical region is pre-calculated by external sensors such as LiDAR and IMU and provided to CA-MOT. If an input image does not have a critical region, the entire frame is considered a critical region. As seen in Fig. 2, GPU is used only for the inference of DNN models, such as the detector (e.g., YOLOv5) and feature extractor (e.g., OSNet), while all other execution is performed on the CPU.

For the association, the MOT system uses the two-step approach [7]. Initially, a CNN-based model (e.g., OS-Net [10]) is employed by the tracker to extract features from the detected objects. The tracker then compares these features between the current and previous frames to identify object pairs with the highest feature similarity. For the remaining objects that are not matched based on feature comparison, a location-based matching method such as IoU (intersection over union) is used. CA-MOT also provides three execution options (i.e., low, middle, and high workloads, respectively) for the association. When it comes to the middle and high workload associations, the tracker extracts features from some (e.g., three) of the detected objects or all of the detected objects, and then performs consecutive feature-based and location-based matchings. On the other hand, low-workload association performs location-based matching only. Depending on the execution option, CA-MOT may extract features from only a subset or all of the detected objects, which means that the feature information of objects may not be updated every time. Therefore, during feature-based matching, the algorithm compares the features extracted from the objects in the current frame with the closest previously extracted features of the tracked objects and matches the two objects with the highest feature similarity.

3.4. Aging-aware task scheduler

The CA-MOT implements a thread-level task scheduler to determine the task to be scheduled and execution options for detection and association for each task at every scheduling decision. The scheduler manages MOT tasks using a single queue and is triggered when a task completes its execution or a new task is released. As three execution options (e.g., low, middle, and high workloads) are provided for each detection and association under CA-MOT, the scheduler decides the image size (e.g., $256 \times 256, 416 \times 416$, and 672×672) for detection and feature size (e.g., zero, three, and more than three) for association according to the scheduling algorithms (to be presented in Section 4).

As discussed in Section 2.3, various combinations of image sizes and the quantity of feature extractions result in different impacts on tracking accuracy. This is due to an important property of the MOT system, which involves supplementing non-updated motion or feature information during detection and association in the current frame by utilizing information from the previous frame. For example, in scenarios with low and middle workload detection, the detection process does not cover the area outside the RoI. Instead, the motion information of objects detected in the previous frame, such as their size, position, velocity, and direction, is leveraged to estimate the corresponding information for objects in the current frame. Moreover, during the association step, if the feature extracted from the immediately preceding frame is unavailable due to low- and middle-workload associations, the feature-based matching algorithm compares the features extracted from objects in the current frame with the features extracted from the nearest past frames. Therefore, the tracking accuracy is determined by the reliability of the reused motion and feature information of the objects. To capture the reliability of the motion and feature information of objects, we propose a new notion of *aging* that specifies the number of middle- or high-workload executions of detection and association conducted from the beginning of the MOT task, respectively. In order to update the motion and feature information as frequently as possible using limited computing resources, it is necessary to balance the aging of detection and association for each task. Note that increasing the aging of detection and association for all tasks simultaneously in every MOT execution is generally not feasible due to limited computing resources. Therefore, a mechanism is required to balance the aging of detection and association for all tasks while providing timing guarantees under constrained resources. To this end, we propose new scheduling algorithms that will be detailed in the next section.

4. Scheduling algorithm

This section presents a task model and proposes new scheduling algorithms building upon CA-MOT.

4.1. Task model

Targeting MOT systems in AVs that involve n camera sensors, we consider a set τ consisting of n MOT tasks denoted as $\tau_i \in \tau$. Each MOT task τ_i is responsible for conducting MOT execution using input images provided periodically by each camera sensor. As we employ the methodology of tracking-by-detection, an MOT task consists of detection and association sub-tasks. Thus, the specification of each MOT task τ_i is given as $\tau_i = (T_i, C_i(s_i, f_i), D_i)$, where T_i represents the period (or the minimum inter-arrival time), $C_i(s_i, f_i)$ denotes the worst-case execution time (WCET) based on the execution options (i.e., low, middle, and high-workload execution) for detection and association sub-tasks, respectively, and D_i indicates the relative deadline. The execution time of the detection sub-task depends on the image size $s_i \in S_i = \{L, M, H\}$, where L, M, H are $256 \times 256, 416 \times 416$, and 672×672, respectively. Note that CA-MOT supports arbitrary non-decreasing sizes for $S_i = \{L, M, H\}$. On the other hand, the execution time of the association sub-task depends on the feature size $f_i \in F_i = \{L, M, H\}$, where L, M, H are zero, from one to three, and more than three, respectively. Note that the tracking-by-detection methodology performs the association phase sequentially through feature-based matching followed by location-based matching using IoU. If f_i is equal

Table 1
Notations used in the scheduling algorithms.

Symbol	Description
τ_i	Task i in the system
T_i	Period of task τ_i (minimum inter-arrival time)
D_i	Relative deadline of task τ_i
$C_i(X, Y)$	Worst-case execution time (WCET) of task i. X: image size for detection (L, M, H) Y: feature size for association (L, M, H)
s_i	Image size for the detection sub-task of task i
f_i	Feature size for the association sub-task of task i
S_i	Set of image size options for task i ($S_i = \{L, M, H\}$)
F_i	Set of feature size options for task i ($F_i = \{L, M, H\}$)
L	Low workload execution
M	Middle workload execution
H	High workload execution
$C_i^D(s_i)$	WCET of the detection sub-task of task i, based on image size s_i
$C_i^A(f_i)$	WCET of the association sub-task of task i, based on feature size f_i
$RC_i(L, L)$	Remaining execution time for the minimum execution of task i
age_i^D	Aging value of the detection sub-task of task i
age_i^A	Aging value of the association sub-task of task i
$slack_{t_{cur}}^i$	Slack time available for task i at the current time t_{cur}
q_i	Minimum execution time of task i in the interval $[t_{cur}, d_1(t_{cur})]$
p	Sum of the minimum execution times for all tasks
$d_1(t_{cur})$	Earliest deadline or future release time at time instant t_{cur}
$slack_i^{D-}$	Remaining slack after executing high-workload detection for task i
$slack_i^{A-}$	Remaining slack after executing high-workload association for task i

to L, this indicates that no feature extraction has been performed for the frame, and thus, feature-based matching is skipped, proceeding directly to location-based matching. In the case of H, we employ the maximum number of objects as defined by the environment (for the dataset considered, this is based on values measured across all videos), for example, 10. Then, the worst-case execution time $C_i(s_i, f_i)$ of each MOT task τ_i is derived as follows.

$$C_i(s_i, f_i) = C_i^D(s_i) + C_i^A(f_i), \tag{1}$$

where $C_i^D(s_i)$ and $C_i^A(f_i)$ are the worst-case execution times of detection and association sub-tasks according to s_i and f_i, respectively. As shown in Fig. 2, both the detection and the association sub-tasks involve GPU operations, with their respective WCETs including the communication costs between the CPU and GPU. Note that the detection sub-task, denoted as τ_i^D, and the association sub-task, denoted as τ_i^A, are executed consecutively without any preemption while sharing the same period and relative deadline. Similarly, when an active task is running, it executes without any interruptions, while other tasks wait in the queue. In addition, each task runs on an environment where non-preemption between the GPU and CPU is guaranteed. To ensure this, while the CPU is running, the GPU waits for input from the CPU. Once the GPU receives the input and is activated, the CPU waits until it receives the results from the GPU, as illustrated in Fig. 2. As seen Fig. 2, GPU is used only for the inference of DNN models, such as the detector (e.g., YOLOv5) and feature extractor (e.g., OSNet), while all other execution is performed on the CPU. Also, CA-MOT does not allow parallel execution for multiple MOT executions (see Table 1).

The worst-case execution time $C_i^D(s_i)$ of the detection sub-task is determined by the sum of various components, including preprocessing time (such as cropping and resizing the input image), image transfer time from CPU memory to GPU memory, model inference time to obtain candidate objects, and postprocessing time (e.g., applying non-maximum suppression) to extract the final objects from the candidates. On the other hand, the worst-case execution time $C_i^A(f_i)$ of the association task depends on the feature size f_i. It is calculated by considering the time required for extracting features from detected objects and performing matching methods such as feature-based and IoU-based matching. An MOT task τ_i is considered *schedulable* if every job J_i (invoked by τ_i) completes its execution within the relative deadline D_i. The overall schedulability of the system is determined by ensuring that every task $\tau_i \in \tau$ is schedulable.

4.2. EDF best-effort

Building upon the system design of CA-MOT presented in Section 3, we develop two scheduling algorithms that aim to provide not only high tracking accuracy for the safety-critical regions but also a timing guarantee for every MOT execution. To this end, the proposed scheduling algorithms have the following two features: (F1) an offline timing guarantee for the minimum execution (i.e., low-workload execution for both detection and association) of every MOT execution and (F2) an online policy to maximize tracking accuracy by systematically increasing workload (i.e., middle- or high-workload execution) of an MOT execution using notions of slack and aging without compromising timing guarantee.

The proposed scheduling algorithms are based on the non-preemptive earliest deadline first (EDF) scheduling algorithm, which assigns higher priority to jobs with earlier deadlines without allowing any preemption. To provide the first feature F1, CA-MOT employs the existing schedulability analysis developed for non-preemptive EDF as follows.

Lemma 1. *For a set τ of MOT tasks scheduled by non-preemptive EDF, minimum execution $C_i(L, L)$ of every task $\tau_i \in \tau$ can be executed without deadline miss as long as the following holds for every task $\tau_i \in \tau$.*

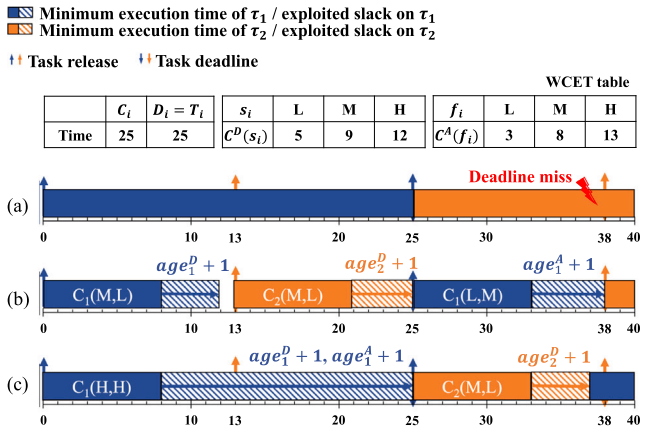
$$\frac{\max_{\tau_i} C_i(L, L)}{\min_{\tau_i} T_i} + \sum_{\tau_i \in \tau} \frac{C_i(L, L)}{T_i} \leq 1.0 \tag{2}$$

Proof. The lemma presents a schedulability condition for non-preemptive EDF, and its proof is outlined as follows. Let us target $\tau_k \in \tau$; also, consider a virtual task $\tau_x \notin \tau$, whose T_x and $C_x(L, L)$ are set to $\min_{\tau_i \in \tau} T_i$ and $\max_{\tau_i \in \tau} C_i(L, L)$, respectively. Now, we compare the finishing time of a job of τ_k when (Case 1) τ is scheduled by non-preemptive EDF, and (Case 2) $\tau \cup \{\tau_x\}$ is scheduled by preemptive EDF. Since at most one lower-priority job can block a high-priority job under non-preemptive scheduling, τ_k can be blocked by at most one lower-priority job under Case 1; obviously, the WCET of the lower-priority job is upper-bounded by $\max_{\tau_i \in \tau} C_i(L, L)$. Also, to block all the following jobs of τ_k, the blocking frequency should be no smaller than T_k, which is lower-bounded by $\min_{\tau_i \in \tau} T_i$. Therefore, the finishing time of a job of τ_i under Case 1 is no later than that under Case 2. Once we apply the well-known schedulability condition for preemptive EDF to Case 2, the condition is the same as Eq. (2), which proves the lemma. □

Note that the proof is self-contained, but a different proof for Lemma 1 can be found in [5,17].

To provide the second feature F2, the proposed scheduling algorithms (i) dynamically increase the workload of each MOT task (e.g., from low workload to middle or high workload) without compromising the timing guarantee while (ii) balance the aging of detection and association of every task. We propose two scheduling algorithms that simultaneously provide (i) and (ii) in different ways: EDF-BE (EDF Best-Effort) and EDF-Slack (EDF with Slack reclamation), adapted from [5]. EDF-BE and EDF-Slack utilize slacks defined differently, but use the same mechanism (in Algorithm 2) to decide on the execution option that employs a notion of aging.

Let $d_1(t_{cur})$ be the earliest deadline or future release time among all tasks at a time instant t_{cur}. The slack $slack_i^{t_{cur}}$ of task τ_i at t_{cur} under the EDF-BE is defined as the expected remaining time up to



	C_i	$D_i = T_i$	s_i	L	M	H	f_i	L	M	H
Time	25	25	$C^D(s_i)$	5	9	12	$C^A(f_i)$	3	8	13

Fig. 3. Execution timeline of multiple MOT tasks under (a) baseline (non-preemptive EDF), (b) EDF-BE, and (c) EDF-Slack scheduling policies.

$d_1(t_{cur})$ after the execution of $C_i(L, L)$ is completed, which is calculated by $d_1(t_{cur}) - t_{cur} - C_i(L, L)$. This slack value is only valid when there are no more than two tasks in the waiting queue at time t_{cur} and no future releases within the interval $[t_{cur}, d_1(t_{cur}))$. Using the slack value conditionally provided at a scheduling decision, EDF-BE can perform middle- or high-workload execution for detection and/or association.

Example. Figs. 3(a) and (b) present a scheduling scenario of the baseline algorithm (i.e., non-preemptive EDF) and EDF-BE with an example task set. We consider an example task set $\tau = \{\tau_1, \tau_2\}$ of which $C_i = C_i^D(H) + C_i^A(H) = 25$, $T_i = 25$, $C_i^D(s_i) = \{5, 9, 12\}$, and $C_i^A(f_i) = \{3, 8, 13\}$ hold for $\tau_i \in \tau$. As shown in Figs. 3(a) and (b), each first job of τ_1 and τ_2 are released at $t = 0$ and $t = 13$, respectively. In the baseline algorithm, the first job of τ_1 executes for 25 time units, and then the first job of τ_2 starts its execution at $t = 25$ resulting in a deadline miss at $t = 38$. Let age_i^D and age_i^A be the aging value of detection and association of τ_i. The aging value is an integer satisfying $age_i^D, age_i^A \geq 0$, and age_i^D and age_i^A for all task $\tau_i \in \tau$ are set to zero at the beginning of the system. The, age_i^D (and age_i^A) increases by one at each time when a detection (and association) is run with middle- or high-workload. In other words, the aging value refers to the number of executions excluding those with low workloads. By adjusting the aging value, a balance is maintained so that neither detection nor association becomes disproportionately large.

Compared to EDF-Slack, EDF-BE is a simpler algorithm that utilizes as many resources as possible, executing a job for greater than $C_i(L, L)$ up to its closest future release only when there is exactly one job in the waiting queue. EDF-BE naturally guarantees no deadline misses in any job execution. This is because, as stated in Lemma 1, the execution of $C_i(L, L)$ without deadline misses for all jobs is guaranteed under EDF. Furthermore, when a job executes for more than $C_i(L, L)$ under EDF-BE, there is only one active job in the waiting queue at that time. In the case of EDF-BE, when the first job of task τ_1 starts its execution at $t = 0$, it executes the minimum execution $C_1(L, L) = 8$ until the earliest deadline or future release at $t = 13$, resulting in a slack of five. Utilizing this slack, the task τ_1 then executes $C_1(M, L)$, and the aging factor age_i^D increases by one. For the first job of task τ_2, released at $t = 13$, there is a slack of 4 until the earliest deadline or future release at $t = 25$. Thus, $C_2(M, L)$ is executed, and age_i^D increases by one. The second job of task τ_1, released at $t = 13$, has a slack of 5 until the earliest deadline or future release at $t = 38$. To balance age_i^D and age_i^A, $C_1(L, M)$ is executed, and age_i^A increases by one. The details of the online policy that effectively balances the aging of detection and association for each task will be provided at the end of this section. As can be observed from the figure, at each scheduling decision, an execution is performed

Algorithm 1 Slack calculation for τ_k at t_{cur} under EDF-Slack

Input: τ, t_{cur}
Output: $slack_k^{t_{cur}}$
1: $p = 0, U =$ the left-hand-side of Equation (2)
2: **for** $i = n$ to $1, \tau_i \in \{\tau_1, ..., \tau_n | d_1(t_{cur}) \leq \cdots \leq d_n(t_{cur})\}$ **do**
3: $\quad U = U - \dfrac{C_i(L, L)}{T_i}$
4: $\quad q_i = \max(0, RC_i(L, L) - (1 - U) \cdot (d_i(t_{cur}) - d_1(t_{cur})))$
5: $\quad U = \min\left(1.0, U + \dfrac{RC_i(L, L) - q_i}{d_i(t_{cur}) - d_1(t_{cur})}\right)$
6: $\quad p = p + q_i$
7: **end for**
8: **return** $slack_k^{t_{cur}} = d_1(t_{cur}) - t_{cur} - p$

that does not exceed the earliest deadline or future release, ensuring execution without deadline misses. The following lemma present the timing guarantee of EDF-BE.

Theorem 1. *A task set τ that satisfies the condition in Eq. (2) is schedulable by EDF-BE.*

Proof. According to Lemma 1, for a task set τ that satisfies Eq. (2), the minimum execution time $C_i(L, L)$ of all tasks $\tau_i \in \tau$ guarantees execution without deadline misses. At each scheduling decision at t under the online policy of EDF-BE, the execution of a job exploiting any slack value does not impose additional inference on any other job. This guarantees that all tasks τ_i receive no more interference than what they would receive under non-preemptive EDF scheduling. Thus, this theorem holds. □

4.3. EDF with slack reclamation

In the case of EDF-BE, more workload than the minimum execution can only be processed when there is a single job in the waiting queue at a given time t_{cur} and no additional releases occur until $d_1(t_{cur})$. This creates a limited opportunity for MOT tasks in CA-MOT to perform more workload than the minimum execution, thus restricting the potential to improve tracking accuracy. To address this limitation, we integrate the approach presented in [5] into EDF-Slack, allowing it to compute slack in a different way than EDF-BE.

Let $d_i(t_{cur})$ denote the ith earliest deadline or release time at t_{cur}, and $RC_i(L, L)$ represent the remaining execution time required to complete the minimum execution $C_i(L, L)$. Algorithm 1 outlines the calculation of the slack value $slack_k^{t_{cur}}$ for task τ_k at t_{cur} within the EDF-Slack algorithm, triggered at each scheduling decision. Since EDF is a job-level fixed-priority scheduling policy, wherein the priority of a job remains constant throughout its execution, scheduling decisions under EDF occur either at the commencement of a job's execution or upon its completion. In the interval $[t_{cur}, d_1(t_{cur})]$, EDF-Slack processes tasks in reverse EDF order, starting from the task with the latest deadline. Job J_k of τ_k has the highest priority at t_{cur}, with $d_1(t_{cur})$ being its deadline, as EDF-Slack follows the EDF policy. The goal of the slack calculation in Algorithm 1 is to delay the execution of all other tasks $\tau_i \in \tau \setminus \tau_k$ beyond $d_1(t_{cur})$ while ensuring that future deadlines are met. This is repeated for all tasks in the waiting queue. To ensure τ_i completes $C_i(L, L)$ before $d_i(t_{cur})$, EDF-Slack calculates the maximum execution time in the interval $d_1(t_{cur}), d_i(t_{cur})$, which is $(1 - U) \cdot (d_i(t_{cur}) - d_1(t_{cur}))$, where U denote the left-hand-side of Eq. (2).

The key steps in the slack calculation are as follows:

- q_i is computed as the minimum execution of τ_i in the interval $t_{cur}, d_1(t_{cur})$ (Lines 3–4).
- $RC_i(L, L)$ is either zero or $C_i(L, L)$, since scheduling decisions are only made upon job completion or release in non-preemptive scheduling (Line 4).

- The execution rate of τ_i in the interval $d_1(t_{cur}), d_i(t_{cur})$ is calculated and recorded (Line 5).
- p is set as the sum of the minimum execution times of all tasks $\tau_i \in \tau$ (Line 6).
- The slack is then determined as the remaining time slots, excluding p (i.e., the sum of q_i), within the interval $t_{cur}, d_1(t_{cur})$ (Line 7).

Example. Fig. 3(c) illustrates a scheduling scenario of EDF-Slack using the same example tasks as shown in Figs. 3(a) and (b). The initial jobs of τ_1 and τ_2 are released at $t = 0$ and $t = 13$, respectively. Applying Algorithm 1, the calculated slack value for τ_1 at $t_{cur} = 0$ is 17, allowing the first job of τ_1 to execute for $C_1(H, H)$ until $t = 25$. Furthermore, age_1^D and age_1^A increment by one. Subsequently, the first job of τ_2 begins its execution at $t = 25$, executing for $C_2(M, L)$ while increasing age_2^D by one. Finally, the second job of τ_2 starts its execution at $t = 37$. Comparing Fig. 3(b) that represents EDF-BE with Fig. 3(c) depicting EDF-Slack, we observe that the aging of τ_1 and τ_2 increases in the same amount in both cases. However, the key difference lies in the execution of the first job of τ_1. Under EDF-Slack, this job is able to execute with a high-workload execution, while under EDF-BE, it can only execute with a middle-workload execution, which allows for higher expectations of tracking accuracy in EDF-Slack.

The following proves the timing guarantee of EDF-Slack.

Theorem 2. *A task set τ that satisfies Eq. (2) is schedulable under EDF-Slack.*

Proof. We prove this by contradiction. Assume, for the sake of contradiction, that the task set τ satisfies Eq. (2), but is not schedulable under EDF-Slack. This implies that at some time t, the total utilization exceeds 1.0, and hence a deadline miss occurs for some job J_i in τ. Let t_{miss} denote the earliest such time at which a deadline miss occurs, i.e., $t_{miss} = d_i$, where d_i is the deadline of J_i. By the definition of EDF-Slack, at each time t, the slack time for each task is computed based on the highest-priority job $J_1(t_{cur})$, where t_{cur} denotes the current time. Since no tasks are released in the interval $[t_{cur}, d_1(t_{cur})]$, the slack time ensures that lower-priority tasks cannot block the execution of J_1. As a result, the blocking term in Eq. (2) remains valid during this interval.

Now, since EDF-Slack is based on EDF scheduling, the total utilization $U(t)$ at any time t can be expressed as:

$$U(t) = \sum_{J_i \in \tau(t)} \frac{C_i}{d_i - t_{cur}} + B(t),$$

where C_i is the remaining execution time of task J_i, and $B(t)$ is the blocking term. According to Eq. (2), $U(t) \leq 1.0$ for all t. Since t_{miss} is the earliest time a deadline miss occurs, we must have $U(t_{miss}) > 1.0$. However, by Eq. (2), we know that $U(t) \leq 1.0$ for all $t \geq t_{cur}$, including t_{miss}. This leads to a contradiction, as the assumption that $U(t_{miss}) > 1.0$ contradicts the fact that $U(t) \leq 1.0$ holds at all times. Therefore, no deadline miss can occur, and the task set τ is schedulable under EDF-Slack. □

Note that the proof is self-contained, but a different proof can be found in [5].

Determination of execution options. EDF-BE and EDF-Slack use different slack concepts to ensure timely execution of tasks while improving tracking accuracy by executing beyond the minimum (i.e., $C_i(L, L)$). As shown in Figs. 3(b) and (c), both EDF-BE and EDF-Slack enhance the aging of detection and association through predefined mechanisms. The goal of these mechanisms is to balance the aging of detection and association, minimizing continuous omissions in updating motion and feature information, thereby maximizing tracking accuracy.

Algorithm 2 outlines the process for determining the execution options for the detection and association steps of task τ_k at time t_{cur} based on the slack $slack_k^{t_{cur}}$ calculated in Algorithm 1.

Algorithm 2 Determination of execution options

Input: τ, t_{cur}, $slack_k^{t_{cur}}$
Output: (s_k, f_k)

1: **if** $slack_k^{t_{cur}} \leq 0$ **then**
2: **return** (L, L)
3: **else**
4: **if** $age_k^D \leq age_k^A$ **then**
5: $slack_k^{D-} = slack_k^{t_{cur}} - (C_k^D(H) - C_k^D(L))$
6: **if** $slack_k^{D-} \geq 0$ **then**
7: **return** $(H, f_k(slack_k^{D-} + C_k^A(L)))$
8: **else**
9: **return** $(s_k(slack_k^{t_{cur}} + C_k^D(L)), L)$
10: **end if**
11: **else**
12: $slack_k^{A-} = slack_k^{t_{cur}} - (C_k^A(H) - C_k^A(L))$
13: **if** $slack_k^{A-} \geq 0$ **then**
14: **return** $(s_k(slack_k^{A-} + C_k^D(L)), H)$
15: **else**
16: **return** $(L, f_k(slack_k^{t_{cur}} + C_k^A(L)))$
17: **end if**
18: **end if**
19: **end if**

- If the slack is less than or equal to zero, the algorithm returns L and L (Lines 1–2).
- Otherwise, the algorithm compares the ages of the detection step (age_k^D) and the association step (age_k^A) (Lines 3–4).
  - If age_k^D is smaller than age_k^A, indicating the detection step requires more resources, the algorithm calculates $slack_k^{D-}$, representing the remaining slack after executing high-workload detection (Line 5).
  - If $slack_k^{D-}$ is greater than or equal to zero, the high-workload detection is followed by middle- or high-workload association depending on $slack_k^{D-}$ (Lines 6–7). In this case, $f_k(x)$ is set as follows:
    - L for $x < C_k^A(M)$,
    - M for $C_k^A(M) \leq x < C_k^A(H)$,
    - H for $x \geq C_k^A(H)$.
  - If $slack_k^{D-}$ is less than zero, the algorithm determines if middle- or high-workload detection can be performed based on $slack_k^{t_{cur}}$, followed by low-workload association (Lines 8–10). In this case, $s_k(x)$ is set as follows:
    - L for $x < C_k^D(M)$,
    - M for $C_k^D(M) \leq x < C_k^D(H)$,
    - H for $x \geq C_k^D(H)$.
- Lines 11–18 follow a similar procedure for determining the execution options, giving preference to the association step. Here, $slack_k^{D-}$ represents the remaining slack after executing the high-workload association.

According to the definition of aging, age_k^D (and age_k^A) increase by one when middle- or high-workload detection (and association) is performed.

DNN-SAM proposed in [10] introduces two scheduling algorithms: EDF-MandFirst and EDF-Slack. Unlike CA-MOT, both DNN-SAM algorithms target multi-object detection (MOD) tasks. The primary distinction between MOT and MOD lies in the presence or absence of dependencies between consecutive frames. In MOD, the detection operation for a given frame does not utilize any information from previous frames. Therefore, techniques that rely on previous frame information,

such as aging-aware methods, cannot be employed in the DNN-SAM algorithms. Another key difference is that DNN-SAM is responsible solely for detection execution and does not handle the association task. Both DNN-SAM and CA-MOT algorithms are based on EDF and prioritize executing jobs with the earliest deadlines among the released tasks. However, in contrast to CA-MOT, DNN-SAM splits each job at release into a mandatory job, responsible for execution in the safety-critical area, and an optional job, responsible for execution in non-critical areas. When any mandatory job is present in the waiting queue, it is always executed first using the EDF algorithm. The distinction between MandFirst and EDF-Slack arises from whether the execution of an optional job may interfere with the execution of a mandatory job. Specifically, the scheduling behavior of DNN-SAM and EDF-Slack operates as follows:

- EDF-MandFirst in [10]: Any mandatory job in the waiting queue has a higher priority than optional jobs and is scheduled using EDF. If no mandatory jobs are in the queue, optional jobs are executed using EDF, ensuring that they do not interfere with the execution of future release jobs of mandatory tasks.
- EDF-Slack in [10]: Any mandatory job in the waiting queue has a higher priority than optional jobs and is scheduled using EDF. If no mandatory jobs are in the queue, optional jobs are executed using EDF, potentially interfering with the execution of future-release mandatory jobs, within the slack calculated from the job's runtime.
- EDF-BE of CA-MOT: A job is not split and has three execution options for both detection and association. It is executed with the maximum workload option to avoid interfering with the execution of future release jobs, but only when exactly one job is present in the waiting queue. The aging of detection and association tasks is considered for accuracy maximization.
- EDF-Slack of CA-MOT: A job is not split and has three execution options for both detection and association. Regardless of the number of jobs in the waiting queue, the job is executed with the maximum workload option, potentially interfering with the execution of future release jobs based on the slack calculated from its runtime. The aging of detection and association tasks is considered for accuracy maximization.

5. Evaluation

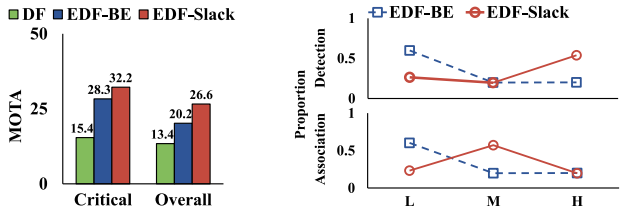
This section evaluates the effectiveness of CA-MOT in achieving R1 and R2 for multiple MOT tasks.

5.1. Experiment setting

- **Software:** CA-MOT employs the tracking-by-detection of which the detector is one of the most popular detectors, YOLOv5 [14] model, and tracker is StrongSORT [7]. We confirmed that other detectors (i.e. YOLOX, Faster-RCNN) exhibit a similar trend to YOLOv5 in terms of MOTA and execution time, as shown in Fig. 7. For feature extraction conducted as a part of association, we used OS-Net [10]. The YOLOv5 model was pretrained on the COCO Dataset [18], while OS-Net was pretrained on the MSMT Dataset [19]. The experimental environment is with Ubuntu 18.04.6 LTS, CUDA 11.4, and PyTorch 1.12.
- **Hardware:** We consider the NVIDIA Jetson Xavier as a GPU-enabled embedded board [20]. The NVIDIA Jetson Xavier features a 64-bit 8-core CPU, 32 GB Memory, and 512-core Volta GPU. We utilized the MAXN mode provided by the NVIDIA Jetson Xavier.
- **Dataset and performance metric:** We used the KITTI Dataset [13], which contains data collected from autonomous vehicle driving. To evaluate the accuracy of each region, we measured the MOTA [15] as the most well-known performance metric for tracking accuracy for critical and entire regions. MOTA compares the ground truth of objects in all frames with the tracking results obtained from the given techniques to measure accuracy. The KITTI dataset consists solely of data captured from forward-facing cameras and does not utilize different cameras, meaning there is no overlap in the areas they cover. Additionally, as it does not assume simultaneous capture by each camera, there are no synchronization issues. CA-MOT aims to maximize the average accuracy for the MOT tasks corresponding to all given cameras without missing any deadlines. This assumes that CA-MOT operates independently of camera interdependencies, with all cameras receiving the same forward-facing camera feed.
- **Execution time measurement:** To obtain the WCET of different execution options for detection and association, we measured the execution time by iterating 1000 times for each sub-tasks with three different execution options of an MOT task and then took the largest value. We also measured the worst-case time required for slack calculation and scheduling decisions such as Algorithms 1 and 2. Table 2 shows the measurement results.

Table 2
Execution time measurement (average and maximum) in terms of image size, feature size, and scheduling overhead.

Time (ms)	C_i^D			C_i^A			C_i^{sche}
	L	M	H	L	M	H	
Average	28.0	30.6	36.7	8.3	63.4	74.4	0.3
Maximum	43.6	53.5	67.6	11.3	74.0	125.2	0.6



(a) Critical and overall accuracy (b) Proportion of execution options

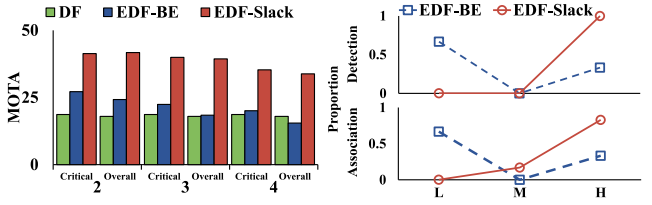
Fig. 4. Comparison for two tasks with the periods (equal to the relative deadlines) of 180 ms and 270 ms.

5.2. Experiment result

We consider task sets in which schedulability is not guaranteed with the high-workload execution for detection and association (i.e., $C_i(H,H)$) for all tasks but is guaranteed with the minimum execution ($C_i(L,L)$) according to Eq. (2). Note that the schedulability with $C_i(x,y)$ for $x,y \in \{L,M,H\}$ can be judged with Eq. (2) by substituting $C_i(L,L)$ to $C_i(x,y)$. To evaluate the effectiveness of CA-MOT we consider the following including a baseline and our two proposed approaches.

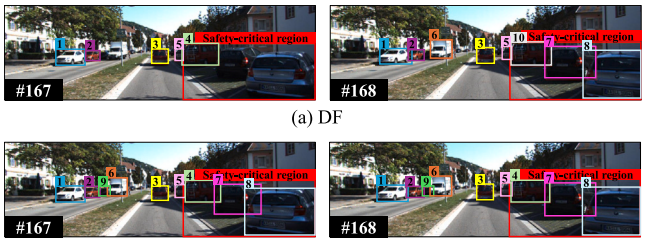
- Detection first (DF): non-preemptive EDF in which the execution option of all tasks $\tau_i \in \tau$ is equally fixed to the rightmost one among $\{C_i(L,L), C_i(M,L), C_i(H,L), C_i(H,M), C_i(H,H)\}$ that satisfies the schedulability condition in Eq. (2).
- EDF-BE: EDF-BE of which task set passes the schedulability condition in Eq. (2), which is proposed in Section 4.2.
- EDF-Slack: EDF-Slack of which task set passes the schedulability condition in Eq. (2), which is proposed in Section 4.3.

Fig. 4 represents the tracking accuracy and the proportion of three execution options (i.e., L, M, and H) selected during detection and association for two tasks with different periods: 180 and 270 ms (milliseconds). As shown in Fig. 4(a), for overall accuracy, EDF-BE and EDF-Slack achieve 20.2% and 26.6%, respectively, while DF achieves



(a) Critical and overall accuracy (b) Proportion of execution options for three tasks case

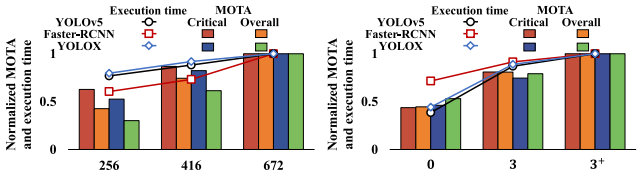
Fig. 5. Comparison for four tasks with the same period (equal to the relative deadline) of 400 ms.



(a) DF

(b) EDF-Slack

Fig. 6. Visualization on KITTI dataset for three tasks with the periods of 400 ms. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)



(a) Size of image (b) Number of feature extraction

Fig. 7. MOTA and execution time on other detectors.

13.4%, which demonstrate the effectiveness of slack utilization and balancing aging of detection and association in increasing tracking accuracy. We observe that the slack reclamation performed by Algorithm 1 in EDF-Slack is significantly more effective in achieving high tracking accuracy than in EDF-BE which has limitations in obtaining a substantial amount of slack. For critical accuracy, EDF-BE and EDF-Slack achieve much higher accuracies, which are 28.3% and 32.2%, respectively, compared to 15.4% of DF. Based on this observation, we can interpret that even though EDF-BE obtains a smaller amount of slack compared to EDF-Slack, it efficiently performs tracking for the critical region with limited computing resources. On the other hand, EDF-Slack provides high tracking accuracy not only for the entire region but also for the safety-critical region, thanks to its efficient slack reclamation. As seen in Fig. 4, EDF-Slack exhibits a significantly higher proportion of high-workload execution and middle-workload execution for detection and association compared to other execution options. On the other hand, EDF-BE shows a slight proportion of middle-workload execution, while the majority of cases involve low-workload execution.

Fig. 5 depicts the results of another experiment involving three different sets of tasks, with the number of tasks ranging from two to four, all having the same periods (i.e., 400 ms with a guaranteed minimum execution $C_i(L, L)$, but no guaranteed maximum execution $C_i(H, H)$ for $\tau_i \in \tau$). In Fig. 5(a), the tracking accuracy of the evaluated approaches is shown as the number of tasks increases. For the case of two tasks, EDF-Slack achieves an overall accuracy of 41.8% and a critical accuracy of 41.4%, while EDF-BE achieves an overall accuracy of 24.3% and a critical accuracy of 27.2%. In contrast, DF achieves lower accuracy, with an overall accuracy of 18.0% and a critical accuracy of 18.7%. As the number of tasks increases, both EDF-BE and EDF-Slack experience a decrease in accuracy, but they still outperform DF in terms of tracking accuracy. Even with only four tasks, EDF-BE yields lower overall accuracy than DF, as it can only detect part of the image when selecting the high workload option. Nevertheless, by prioritizing computations in critical regions at low and medium workloads, EDF-BE attains higher critical accuracy than DF. Fig. 5(b) presents the distribution of execution options for EDF-BE and EDF-Slack when there are three tasks. Similar to Fig. 4, it is evident that both EDF-BE and EDF-Slack allocate the workload between the detection and association steps in a balanced manner using the ages. Additionally, EDF-Slack can reclaim more slack compared to EDF-BE.

Fig. 6 presents the tracking outcomes of a single task within a set of three tasks, each with a period of 400 ms, comparing (a) the DF algorithm and (b) EDF-Slack. In the visualization, each tracked object is represented by a unique color and ID within a bounding box, with the symbol "#" indicating the frame number. In the DF scenario, each task executes $C_i(H, L)$, leading to insufficient computational resources for proper association. This inadequacy results in DF's failure to track two objects within the safety-critical region in the 167th frame and causes an ID switch from 4 to 10 in the subsequent 168th frame, as illustrated in Fig. 6. Conversely, EDF-Slack leverages aging and slack techniques to allocate sufficient computational resources for both detection and association tasks, enabling accurate tracking of all objects in the safety-critical region.

Additional experiments are conducted to ascertain if CA-MOT exhibits comparable behavioral patterns across a range of detectors, including YOLOv5, which was evaluated previously. Fig. 7 displays the MOTA and execution time for various contemporary detectors, analyzed according to their workload. Modern detectors are generally classified into one-stage and two-stage categories based on their architecture and further into anchor-free and anchor-based types, contingent on their use of predefined anchors for object detection. Our study incorporated YOLOv5, a standard one-stage anchor-based detector. We also investigate the performance of the two-stage anchor-based detector Faster-RCNN [9] and the one-stage anchor-free detector YOLOX [21], to verify the consistency of results. Faster-RCNN utilized ResNet-50 as its backbone network, while YOLOX was configured with a small version model. Both models were trained using the COCO dataset. Despite minor discrepancies in specific ratios, the results consistently demonstrate that both MOTA and execution time escalate in conjunction with increasing workload, as shown in Figs. 7(a) and (b). The runtime trend of YOLOX is particularly noteworthy, which closely mirrors that of YOLOv5. This pattern indicates that similar outcomes may be expected from other detectors akin to YOLOv5.

6. Related work

The tracking-by-detection model is a commonly used method in the MOT field. It has shown significant progress and enhanced performance recently, largely thanks to the evolution of deep neural networks (DNNs). A well-recognized model in this field, SORT (simple online and real-time tracking) [22], does its matching based mainly on where objects are located, using detection tools to achieve this. To push this model further, DeepSORT [6] builds on the SORT model by adding a DNN-based re-identification model. This allows for the extraction of object features. By adding this layer, DeepSORT utilizes both the object's location and its visual information, leading to a stronger performance. Recent work in this area, such as Deep OC-SORT [23] and StrongSORT [7], is geared towards enhancing the accuracy of these models even more, focusing especially on refining and improving the matching algorithms used in these systems. However, it is critical to understand that these approaches are mainly designed for situations where there are plenty of computing resources. Therefore, they might struggle to meet the timing needs in systems that are restricted in resources, like

the embedded systems in self-driving vehicles where resources may be scarce.

Considering self-driving vehicles, which are fundamentally systems where safety is critical, even the smallest delays or slight drops in accuracy can lead to significant and potentially dangerous risks. Some research such as DNN-SAM [5] has tried to tackle these problems by suggesting frameworks that concentrate specifically on safety-critical areas. These frameworks give priority to critical accuracy and use uncertainty handling to ensure the highest safety standards. However, these research studies and their related approaches are mostly designed for multi-object detection systems and may not directly apply to or be effective in multi-object tracking. Likewise, another study, RT-MOT [2], aims to maximize the overall accuracy of multi-object tracking and ensure on-time execution, but it overlooks the importance of individual objects in its approach. To address these limitations, our suggested framework, known as CA-MOT, aims to confront these challenges directly. By leveraging the unique traits of multi-object tracking in safety-critical systems, CA-MOT ensures on-time execution and boosts tracking accuracy for objects that could potentially be dangerous to the system. It builds on previous work while addressing their weaknesses to create a safer and more efficient tracking system.

7. Discussion

A limitation of CA-MOT is its exclusive reliance on a single CPU and GPU, which restricts scalability. A recent approach, Batch-MOT [3], addresses this limitation by processing input images from multiple cameras through a shared queue, distributing CPU operations across multiple CPUs, and employing batch processing on a single GPU. However, this approach may introduce additional communication overhead among CPUs, potentially determining its overall efficiency. The primary contribution of Batch-MOT lies in its online schedulability analysis, which dynamically determines the maximum number of images that can be batch-processed without violating their deadlines. Nonetheless, unlike CA-MOT, Batch-MOT lacks support for multiple execution strategies during the association phase, resulting in suboptimal resource utilization for individual MOT tasks. Enhancing CA-MOT by incorporating batch processing capabilities to address these shortcomings presents a promising avenue for future research. Furthermore, as highlighted in previous studies, deploying CA-MOT on real-world platforms, such as the F1/10 autonomous driving platform [5], offers significant potential for further investigation and practical validation.

8. Conclusion

In this paper, we proposed, CA-MOT, a new criticality-aware MOT execution and scheduling framework. Aiming at achieving critical-accuracy maximization and timing guarantee, CA-MOT first proposes a new system design to offer a control knob between tracking accuracy and timing guarantee to efficiently utilize limited computing resources. Then, CA-MOT develops two scheduling algorithms to effectively utilize the system design while using the notions of slack and aging of detection and association. Using various task sets and real-world autonomous driving data, we demonstrated that CA-MOT can obtain high tracking accuracy of entire and safety-critical regions while ensuring the timely execution of all MOT tasks.

CRediT authorship contribution statement

Donghwa Kang: Writing – original draft, Software, Methodology, Formal analysis. **Jinkyu Lee:** Writing – review & editing, Validation, Formal analysis. **Hyeongboo Baek:** Writing – review & editing, Supervision, Funding acquisition, Formal analysis, Conceptualization.

Declaration of competing interest

The authors declare the following financial interests/personal relationships which may be considered as potential competing interests: Hyeongboo Baek reports financial support was provided by National Research Foundation of Korea. If there are other authors, they declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Acknowledgments

This work was supported by the National Research Foundation of Korea (NRF) grant funded by the Korea government (MSIT) (RS-2023-00250742, 2022R1A4A3018824, RS-2024-00438248). This work was partly supported by the Institute of Information & Communications Technology Planning & Evaluation(IITP)-ITRC(Information Technology Research Center) grant funded by the Korea government(MSIT) (IITP-2025-RS-2023-00259061).

Data availability

Data will be made available on request.

References

[1] M. Yang, S. Wang, J. Bakita, T. Vu, F.D. Smith, J.H. Anderson, J.-M. Frahm, Re-thinking CNN frameworks for time-sensitive autonomous-driving applications: Addressing an industrial challenge, in: Proceedings of IEEE Real-Time Technology and Applications Symposium, IEEE, 2019, pp. 305–317.

[2] D. Kang, S. Lee, H.S. Chwa, S.-H. Bae, C.M. Kang, J. Lee, H. Baek, RT-MOT: Confidence-aware real-time scheduling framework for multi-object tracking tasks, in: Proceedings of IEEE Real-Time Systems Symposium, IEEE, 2022, pp. 318–330.

[3] D. Kang, S. Lee, C.-H. Hong, J. Lee, H. Baek, Batch-MOT: Batch-enabled real-time scheduling for multi-object tracking tasks, IEEE Trans. Comput.-Aided Des. Integr. Circuits Syst. (2024).

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

[5] W. Kang, S. Chung, J.Y. Kim, Y. Lee, K. Lee, J. Lee, K.G. Shin, H.S. Chwa, DNN-SAM: Split-and-merge DNN execution for real-time object detection, in: Proceedings of IEEE Real-Time Technology and Applications Symposium, 2022, URL https://rtcl.dgist.ac.kr/index.php/publication-2/.

[6] N. Wojke, A. Bewley, D. Paulus, Simple online and realtime tracking with a deep association metric, in: Proceedings of the IEEE International Conference on Image Processing, IEEE, 2017, pp. 3645–3649.

[7] Y. Du, Y. Song, B. Yang, Y. Zhao, StrongSORT: Make deepsort great again, 2022, arXiv preprint arXiv:2202.13514.

[8] J. Redmon, S. Divvala, R. Girshick, A. Farhadi, You only look once: Unified, real-time object detection, in: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, 2016, pp. 779–788.

[9] R. Girshick, Fast R-CNN, in: Proceedings of the IEEE International Conference on Computer Vision, 2015, pp. 1440–1448.

[10] K. Zhou, Y. Yang, A. Cavallaro, T. Xiang, Omni-scale feature learning for person re-identification, in: Proceedings of the IEEE International Conference on Computer Vision, 2019, pp. 3702–3712.

[11] Y. Zhang, P. Sun, Y. Jiang, D. Yu, F. Weng, Z. Yuan, P. Luo, W. Liu, X. Wang, ByteTrack: Multi-object tracking by associating every detection box, in: Proceedings of the European Conference on Computer Vision, Springer, 2022, pp. 1–21.

[12] Y. Zhang, C. Wang, X. Wang, W. Zeng, W. Liu, FairMOT: On the fairness of detection and re-identification in multiple object tracking, Int. J. Comput. Vis. 129 (11) (2021) 3069–3087.

[13] A. Geiger, P. Lenz, R. Urtasun, Are we ready for autonomous driving? The KITTI vision benchmark suite, in: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, 2012.

[14] ultralytics, YOLOv5 [Online], 2022, Available: https://github.com/ultralytics/yolov5.

[15] K. Bernardin, R. Stiefelhagen, Evaluating multiple object tracking performance: the clear mot metrics, EURASIP J. Image Video Process. 2008 (2008) 1–10.

[16] G. Welch, G. Bishop, et al., An introduction to the Kalman filter, ACM SIGGRAPH (1995).

[17] T.P. Baker, A stack-based resource allocation policy for realtime processes, in: Proceedings of IEEE Real-Time Systems Symposium, IEEE, 1990, pp. 191–200.
[18] T.-Y. Lin, M. Maire, S. Belongie, J. Hays, P. Perona, D. Ramanan, P. Dollár, C.L. Zitnick, Microsoft COCO: Common objects in context, in: Proceedings of the European Conference on Computer Vision, Springer, 2014, pp. 740–755.
[19] L. Wei, S. Zhang, W. Gao, Q. Tian, Person transfer gan to bridge domain gap for person re-identification, in: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, 2018, pp. 79–88.
[20] NVIDIA, NVIDIA Xavier Developer Kit. [Online], 2022, Available: https://www.nvidia.com/en-us/autonomous-machines/embedded-systems/jetson-agx-xavier.
[21] Z. Ge, S. Liu, F. Wang, Z. Li, J. Sun, Yolox: Exceeding yolo series in 2021, 2021, arXiv preprint arXiv:2107.08430.
[22] A. Bewley, Z. Ge, L. Ott, F. Ramos, B. Upcroft, Simple online and realtime tracking, in: Proceedings of the IEEE International Conference on Image Processing, IEEE, 2016, pp. 3464–3468.
[23] G. Maggiolino, A. Ahmad, J. Cao, K. Kitani, Deep oc-sort: Multi-pedestrian tracking by adaptive re-identification, 2023, arXiv preprint arXiv:2302.11813.

Dongwha Kang is a Ph.D. course student in the School of Computing, Korea Advanced Institute of Science and Technology (KAIST), South Korea. He received a BS and MS degree in computer science from Incheon National University (INU) in 2022 and 2024 respectively. His research interests include artificial intelligence, autonomous systems, and real-time embedded systems.

Jinkyu Lee is an associate professor in the Department of Computer Science and Engineering at Sungkyunkwan University (SKKU), South Korea, where he joined in 2014. He received the BS, MS, and Ph.D. degrees in computer science from the Korea Advanced Institute of Science and Technology (KAIST), South Korea, in 2004, 2006, and 2011, respectively. He has been a research fellow/visiting scholar in the Department of Electrical Engineering and Computer Science, University of Michigan until 2014. His research interests include system design and analysis with timing guarantees, QoS support, and resource management in real-time embedded systems and cyber–physical systems. He won the best student paper award from the 17th IEEE Real-Time and Embedded Technology and Applications Symposium (RTAS) in 2011 and the Best Paper Award from the 33rd IEEE Real-Time Systems Symposium (RTSS) in 2012.

Hyeongboo Baek is an associate professor in the Department of Artificial Intelligence, University of Seoul (UOS), South Korea. He received the BS degree in Computer Science and Engineering from Konkuk University, South Korea, in 2010 and the MS and Ph.D. degrees in Computer Science from KAIST, South Korea, in 2012 and 2016, respectively. His research interests include cyber–physical systems, real-time embedded systems, and system security. He won the best paper award from the 33rd IEEE Real-Time Systems Symposium (RTSS) in 2012.