MOT-AS: Real-Time Scheduling Framework for Multi-Object Tracking Capturing Accuracy and Stability

Donghwa Kang
Incheon National University
Dept. of Computer Science and Engineering
Incheon, Republic of Korea
anima0729@inu.ac.kr

Kilho Lee
Soongsil University
School of AI Convergence
Seoul, Republic of Korea
khlee.cs@ssu.ac.kr

Cheol-Ho Hong
Chung-Ang University
Dept. of Intelligent Semiconductor Engineering
Seoul, Republic of Korea
cheolhohong@cau.ac.kr

Youngmoon Lee
Hanyang University
Dept. of Robotics
Ansan, Republic of Korea
youngmoonlee@hanyang.ac.kr

Jinkyu Lee*
Sungkyunkwan University
Dept. of Computer Science and Engineering
Suwon, Republic of Korea
jinkyu.lee@skku.edu

Hyeongboo Baek†
Incheon National University
Dept. of Computer Science and Engineering
Incheon, Republic of Korea
hbbaek@inu.ac.kr

ABSTRACT

Unlike existing accuracy-centric multi-object tracking (MOT), MOT subsystems for autonomous vehicles (AVs) must *accurately* perceive the surrounding conditions of the vehicle *and* timely deliver the perception results to the control subsystems before losing *stability*. In this paper, we proposed MOT-AS (Multi-Object Tracking systems capturing Accuracy and Stability), a novel handover-aware MOT execution and scheduling framework tailored for AVs with multi-cameras, which aims to maximize tracking accuracy without sacrificing system stability. Given the resource limitations inherent to AVs, MOT-AS partitions the handover-aware MOT execution into two distinct sub-executions: tracking handover objects that move across multiple cameras (referred to as global association) and those that move within a single camera (termed local association). It selectively performs the global association only when necessary and carries out local association with multiple execution options to explore the trade-off between accuracy and stability. Building upon MOT-AS, we developed a new scheduling framework encompassing a new MOT task model, offline stability analysis, and online scheduling algorithm to maximize accuracy without compromising stability. We implemented MOT-AS on both high-end and embedded GPU platforms using the Nuscenes dataset, demonstrating enhanced tracking accuracy and stability over conventional MOT systems, irrespective of their handover considerations.

*Jinkyu Lee is the co-corresponding author of this paper.
†Hyeongboo Baek is the corresponding author of this paper.

SAC '24, April 8–12, 2024, Avila, Spain

ACM ISBN 979-8-4007-0243-3/24/04…$15.00
https://doi.org/10.1145/3605098.3635996

CCS CONCEPTS

• **Computer systems organization** → **Real-time operating systems**; **Embedded software**; • **Computing methodologies** → **Tracking**; **Object detection**.

KEYWORDS

Autonomous vehicles, multi-object tracking, handover, stability analysis, real-time scheduling

ACM Reference Format:
Donghwa Kang, Kilho Lee, Cheol-Ho Hong, Youngmoon Lee, Jinkyu Lee, and Hyeongboo Baek. 2024. MOT-AS: Real-Time Scheduling Framework for Multi-Object Tracking Capturing Accuracy and Stability. In *The 39th ACM/SIGAPP Symposium on Applied Computing (SAC '24), April 8–12, 2024, Avila, Spain.* ACM, New York, NY, USA, Article 4, 10 pages. https://doi.org/10.1145/3605098.3635996

1 INTRODUCTION

A modern autonomous vehicle (AV), serving as a compelling example of cyber-physical systems (CPSes), periodically engages in deep neural network (DNN)-based perception tasks coupled with decision-driven physical control tasks [4]. Perception information (e.g., detected objects) must be highly accurate and provided in a timely manner to control tasks for proper decision-making, such as obstacle avoidance [4]. If perception information continually fails to be conveyed in time, AV's stable control for obstacle avoidance can be lost due to applying outdated perception information, potentially leading to a catastrophic accident within the system. Therefore, the perception subsystem of AVs carries a crucial requirement: it must (G1) *accurately* perceive the surrounding conditions of the vehicle and (G2) timely deliver the perception results to the control subsystem before losing *stability*.

The DNN-based multi-object tracking (MOT) is a representative perception subsystem deployed in modern AVs, which tracks objects (e.g., cars and pedestrians) around the vehicle through multiple cameras and periodically delivers this information to the control subsystem [4]. There are key characteristics of multi-camera-based MOT systems: an object being tracked in the view of one camera

moves into the view of another camera called *handover*, and such handover occurs conditionally. When tracking a normal object, as opposed to a handover object, the comparison occurs between consecutive frames from the same camera, eliminating the need to reference other cameras. For instance, Figure 1(b) depicts that tracking objects ① and ② are computed exclusively for the front camera while tracking object ③ is confined to the side camera's computations. Conversely, the occasional appearance of a handover object demands additional computation, as it requires cross-referencing data from multiple cameras simultaneously. For instance, in Figure 1(c), to track the handover of object ② as it moves from the view of the front camera to that of the side camera, it is necessary to associate its presence in the front camera at t+1 with its appearance in the side camera at t+2.

Since AVs provide limited computing resources as representative embedded systems, performing heavy computation to achieve G1 in every MOT task can compromise G2 due to continuous deadline misses, which is, a challenge that existing handover-aware approaches fail to address without consideration of resource constraints [5, 12, 13]. Our strategy to address this challenge involves decomposing the handover-aware MOT execution into (i) tracking handover objects and (ii) tracking normal (non-handover) objects, which have distinct characteristics. The computation for (i) is conditionally required and executed only when necessary by capturing the handover moments. Meanwhile, (ii) is always necessary as long as the detected object exists. Thus, we offer various execution options that provide a trade-off between G1 and G2 for (ii), allowing for selective execution based on the available computing resources. Tracking normal and handover objects is equally crucial to achieving G1, so handover-aware MOT systems on limited resources have the following requirements.

R1. It should capture the timing of handover to track handover objects only if necessary and offer various execution options to provide the trade-off between G1 and G2 for tracking normal objects.

Based on the perception information received from perception subsystems, the control subsystems of modern AVs strive to maintain stability by periodically executing control tasks. Existing real-time perception task model [4] presupposes that all perception tasks must complete their execution and deliver them to the control subsystems before their deadlines. This strict requirement can often be excessively cautious or pessimistic as actual physical systems have inbuilt safety margins and thus do not lose stability even if some jobs miss their deadlines [7]. For instance, the control subsystem based on MOT systems might tolerate occasional misses in perception information updates, as it can utilize perception information received in the previous period before losing system stability [6]. Allowing such sporadic misses in updating perception information enables higher utilization of computing resources by accepting more diverse execution schedules. However, to efficiently utilize these advantages in a handover-aware MOT system to achieve G1 and G2, a technique must first be developed that allows the target MOT system to verify G2 is guaranteed at design time; this process is known as *offline stability analysis*. Subsequently, among the scheduling options that satisfy G2, the best schedule of perception information in terms of accuracy must be found using the answer to R1 at every MOT execution to achieve G1; this process is referred to as the *online scheduling algorithm*. These strategies underline that a handover-aware MOT system aiming to achieve G1 and G2 has the following requirement.

R2. It should develop offline stability analysis and an online scheduling algorithm that maximizes tracking accuracy while ensuring stability, using the interface provided by the answer to R1.

In this paper, we propose MOT-AS (Multi-Object Tracking systems capturing Accuracy and Stability), a novel handover-aware MOT execution and scheduling framework designed for AVs equipped with multi-cameras, which aims to maximize tracking accuracy without sacrificing system stability. To address R1, MOT-AS introduces a new system design that handles as many MOT tasks as cameras in AVs, constantly updating the state of the objects being tracked by each MOT task, to be detailed in Section 2.1. By comparing the states of objects across different MOT tasks, MOT-AS can estimate the timing of handovers, referred to as *handover identification*, and selectively execute the tracking of handover objects, known as *global association*. Moreover, MOT-AS incorporates the notion of a *confidence threshold* to determine the number of objects participating in tracking, to be detailed in Section 2.2. This enables two distinct execution options, named *low-workload* and *high-workload*, to track normal objects, referred to as *local association*, offering the trade-off between tracking accuracy (affecting G1) and execution time (affecting G2). In essence, MOT-AS basically offers two execution alternatives for the local association, one with the global association and the other without, resulting in four execution options for each MOT task.

To fulfill R2, MOT-AS implements a new scheduling framework that leverages the system design developed to address R1. To this end, we first propose a new MOT task model by expanding the existing stability-aware fault-tolerant CPS task model to properly represent the multiple MOT execution options offered by the answer to R1. Then, motivated by the limited resources provided by AVs, we target MOT systems where a high-workload local association along with a global association for every MOT execution cannot conserve the stability of the system. For the target systems, we propose a new offline stability analysis that ensures the minimum execution of every MOT task (i.e., a low-workload local association without global association) while providing stability. Building upon the MOT task model and offline stability analysis, we develop an online scheduling algorithm, NPFP$^{\text{ML}}$ (non-preemptive fixed-priority with mixed local and global associations) that assesses the feasibility of the diverse execution options provided by the answer to R1 and then selects the best choice that maximizes accuracy without undermining the stability verified by the offline analysis.

We deployed MOT-AS on both high-end and embedded GPU platforms using the Nuscenes as a handover-aware dataset, demonstrating superior tracking accuracy and stability over traditional MOT systems, regardless of their handover considerations.

Our contributions are as follows.

- We raise the motivation of systematically decomposing handover-aware MOT execution into the global and local associations to achieve G1 and G2 in resource-limited AVs (Section 2).

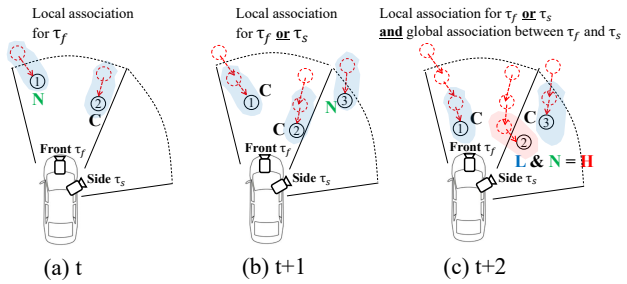


Figure 1: Different states of objects ("N"ew, "C"onfirmed, "L"ost, "H"andover) tracked by two MOT tasks for a front camera τ_f and a side camera τ_s, respectively

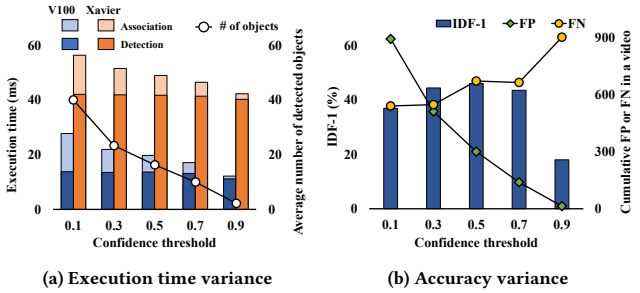


Figure 2: Differences in execution time and accuracy for various confidence thresholds in MOT execution

- We propose a new system design MOT-AS to execute global association selectively and provide a trade-off between G1 and G2 for local association (Sections 3 and 4).
- Building upon MOT-AS, we implement a scheduling framework that maximizes accuracy while providing a stability guarantee (Section 5).
- We demonstrate the effectiveness of MOT-AS through experiments on both high-end and embedded GPU computing systems (Section 6).

2 TARGET SYSTEM AND MOTIVATION

2.1 Target system

We target handover-aware MOT systems that operate on autonomous vehicles equipped with n cameras. Each camera can operate at different frame-per-second (FPS) due to different uses (e.g., forward-facing cameras operate at higher rates, while side-facing cameras typically work at lower rates). MOT-AS adopts a *tracking-by-detection* method, where front-end detection and back-end association are consecutively performed for each frame continuously inputted from each camera. For detection, a DNN-based stand-alone detector (e.g., YOLO series [14] and R-CNN series [15]) is deployed to identify the location and class of objects in an input frame. Then, a local association is executed to match the two objects with the highest similarities from consecutive frames of a single camera. Subsequently, a global association is carried out to identify an object of a handover between two adjacent cameras.

2.2 Motivation

Figure 1 presents the different states of objects that can be tracked in a multi-camera-based MOT system. For an MOT task τ_i corresponding to a camera and the t-th frame that the task processes, an object can have four different states: *new*, *confirmed*, *lost*, and *handover* (denoted by N, C, L, and H, respectively). Figure 1 illustrates an example of objects with different states (i.e., N, C, L, and H) in three consecutive frames t, $t+1$, and $t+2$, being tracked by two MOT tasks τ_f and τ_s for the front camera and the side camera of an AV, respectively. In the figure, each circle presents an object with an identification (ID) number in the circle. For the t-th frame of τ_f, the object ① is referred to as a new object that is detected for the first time by τ_f. Confirmed objects are those consecutively detected a specific number of times (e.g., twice in Figure 1) by the camera, such as an object ②. In the $(t+1)$-th frame, ① is again detected by τ_f, becoming a confirmed object. Furthermore, a new object ③ is detected by τ_s, making it a new object to τ_s. In the $(t+2)$-th frame, ③ is detected once more, becoming a confirmed object, and ② becomes a handover object, being a lost object to τ_f but a new object to τ_s. From Figure 1, we have the following observation.

O1. It is possible to capture the timing of a handover by comparing the states of objects in adjacent cameras, and a global association can be selectively performed only when a handover occurs.

To provide a trade-off between accuracy and execution times for a local association, we focus on the notion of *confidence threshold* used in the detector's detection. A multi-object detector extracts candidate objects before determining the detected objects' class (e.g., car or pedestrian) and location. These candidate objects have a confidence value, indicating the likelihood of belonging to each class. For instance, if an object signifies the pedestrian class with a confidence of 0.9, it means that the detector model perceives that object as a pedestrian with 90% certainty. If the confidence threshold is set at 0.9, it means that only those candidate objects with confidence exceeding 0.9 will be presented as the final detection result, and only these objects will become the targets for the association.

Figures 2(a) and (b) illustrate the variation in execution time and tracking accuracy, respectively, in accordance with different confidence thresholds. The experiment results in the figures are obtained using the NuScences dataset [2], on a high-end GPU (i.e., NVIDIA Tesla V100 [8] comparable to NVIDIA Orin system-on-chip (SoC) providing similar GPU capability for Tensor Core operations [10]) and a GPU embedded board (i.e., NVIDIA Jetson Xavier system-on-chip (SoC) [9]).

A total of six cameras were considered (targeting the front, rear, front-right, front-left, rear-right, and rear-left); YOLOv5 [17] and OC-SORT [3] are employed as the detector and tracker, respectively. As depicted in Figure 2(a), the higher the confidence threshold, the time assigned to the association diminishes because the average number of objects partaking in the association reduces (e.g., from 40 to 1.5 for confidence thresholds of 0.1 to 0.9).

Figure 2(b) represents the fluctuating accuracy contingent on the alteration in the confidence threshold. We employed IDF-1 as

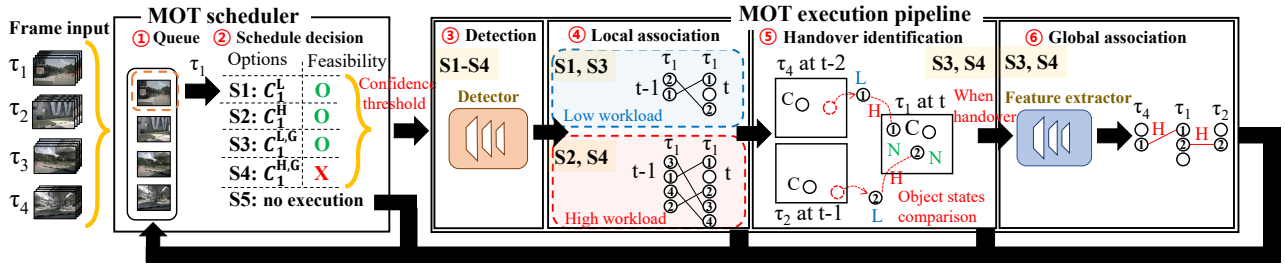


Figure 3: System design of MOT-AS

the accuracy metric, commonly utilized for handover-aware MOT systems [16]. Within the figure, FP denotes false positives, signifying the count of objects mistakenly perceived as existing when they do not, while FN refers to false negatives, symbolizing the count of objects that are indeed present but remain undetected. As observed in the figure, the apex of tracking accuracy is marked when the confidence threshold is at 0.5. This pattern emerges because when the confidence threshold is low, an excess of objects is detected compared to what actually exists, thereby augmenting FP. Conversely, when the confidence threshold is high, a deficit of objects are detected relative to what exists, giving rise to an escalation in FN. Further particulars about IDF-1 are elaborated in Section 6.

Through Figures 2(a) and (b), it becomes evident that an overly low confidence yields diminished accuracy relative to execution time. Hence, after pinpointing the best confidence threshold with respect to accuracy (e.g., 0.5 in Figure 2(b)), increasing the confidence threshold reduces execution time but concurrently decreases accuracy. This dynamic facilitates a trade-off between the two, leading to subsequent observation.

O2. Carefully selecting the range of the confidence threshold provides a reasonable trade-off between tracking accuracy and execution time for a local association.

3 MOT-AS: SYSTEM OVERVIEW

MOT-AS provides an MOT execution and scheduling framework for a handover-aware MOT system operating in AVs, where high accuracy should be achieved for both normal and handover objects while providing stability under limited computing resources. To address R1 and R2 in Section 1, the design of MOT-AS should consider the following design issues.

I1. It should design the system architecture that captures the timing of handovers for selectively tracking handover objects and provides different execution options for tracking normal objects (using O1 and O2 in Section 2.2 to address R1).

I2. It should maximize tracking accuracy while enabling stability guarantee using the answer of I1 (to address R2).

To address I1, MOT-AS develops an MOT execution pipeline based on the tracking-by-detection methodology. Based on the scheduling decisions of the MOT scheduler, the MOT execution pipeline sequentially performs the followings:

- **Detection** to localize and classify objects in an input image,
- **Local association** to track normal objects with two different execution options offering a trade-off between execution times and accuracy,
- **Handover identification** to capture the timing of handover, and
- **Global association** to selectively track handover objects,

which are detailed in Section 4. These executions are performed in a single process and alternate with the thread-level MOT scheduler. To address I2, the MOT scheduler utilizes the followings:

- **Task model** for representing the parameters of MOT execution, essential for both offline stability analysis and the online scheduling algorithm, to be detailed in Section 5.1,
- **Offline stability analysis** to provide stability guarantee before run-time, to be detailed in Section 5.2, and
- **Online scheduling algorithm** that determines the execution options of local association with or without global association by identifying the available computing resources, to be detailed in Section 5.3.

Both the MOT execution pipeline and the thread-level MOT scheduler communicate using shared memory to exchange the relevant information for MOT execution and schedule (e.g., scheduling decisions, task parameters, and object states).

Workflow. Figure 3 depicts the workflow of MOT-AS. Every MOT task periodically acquires frame images from its designated camera, subsequently generating jobs to be in the queue (①). Using an online scheduling algorithm, the MOT scheduler assigns priority to MOT tasks and determines the execution options (low- or high-workload local association with the possibility of global association) for the highest-priority task (②). Following this decision, the MOT execution pipeline first conducts detection using a pre-deployed detector (③). The local association employs the detected results from the current frame and performs IoU-based matching with objects of the preceding frame (④). To identify handovers, the states of objects in both the current and former frames of neighboring cameras are compared (⑤). If a handover is detected and resources are available for a global association, features from the handover objects are extracted, setting the stage for feature-based matching (⑥); otherwise, a global association is not conducted. Once all matches are determined, the tracking results are passed to the control subsystem, preparing the execution of the subsequent MOT task.

4 MOT EXECUTION PIPELINE

The MOT execution pipeline in Figure 3 supports four MOT execution steps: detection, local association, handover identification, and global association.

Detection. In detection, the locations and classes (e.g., cars and pedestrians) of objects in the designated input image are identified. MOT-AS employs any existing stand-alone DNN-based detector (e.g., YOLO and R-CNN series [14, 15]) for object detection. The detector first extracts candidate objects (in the input image) of which confidence values are from 0 to 1, reflecting its likelihood of belonging to a particular class. We focus on the *confidence threshold* used in the non-maximum suppression (NMS) process for extracted candidate objects, which is adopted by most (if not all) DNN-based detectors to finalize the detection results among candidate objects. The NMS filters objects based on a dynamically adjustable confidence threshold parameter, thereby determining the number of detected objects, as discussed in Section 2.2. Once the MOT scheduler determines a confidence threshold (eventually determining the workload of the local association), it is used for the deployed detector to set its confidence threshold. Note that we only dynamically adjust the confidence threshold and make no other modifications to the existing detector.

Local association. For an MOT task τ_i, local association matches the detected objects of the current frame (e.g., at t) with the objects from the previous frame (e.g., t-f ($f > 0$) at which the most recent execution occurs before t). MOT-AS employs any existing tracker (e.g., SORT [1], ByteTrack [18], and OC-SORT [3]) that supports IoU-based matching for the association (and feature-based matching to be used for global association). Depending on the given confidence threshold, the local association can be performed in two modes: *low-workload* and *high-workload*. The low-workload local association prioritizes reduced execution time over accuracy. In this mode, only objects detected in the current frame with a high confidence threshold (e.g., 0.9) or above and those being tracked in the previous frame are considered for association. On the other hand, the high-workload local association emphasizes accuracy at the expense of longer execution times. In this mode, only objects detected in the current frame with a low confidence threshold (e.g., 0.5) or above and those tracked in the previous frame are included in the association. Note that the confidence threshold for low- and high-workload depends on the deployed detector model and dataset and is determined emperically. Once the association is completed, the states (i.e., new, confirmed, lost, or handover, as discussed in Section 2.1) of the objects are updated, and a unique ID is assigned to each object.

Hanover identification. Consider an MOT task τ_i that performs local association on the current frame (e.g., at t), and two tasks $\tau_{(i-1)}$ and $\tau_{(i+1)}$ of cameras physically adjacent to the camera of τ_i, respectively. To identify the timing of handovers, MOT-AS compares the object states (i.e., new, confirmed, lost, and handover discussed in Section 2.1) from the t-th frame of τ_i with the states from the (t-f)-th and (t-g)-th frames of $\tau_{(i-1)}$ and $\tau_{(i+1)}$, respectively, where t-f (f>0) and t-g (g>0) are the most recent execution occurs before t for $\tau_{(i-1)}$ and $\tau_{(i+1)}$, respectively. If a lost object is detected in $\tau_{(i-1)}$ or $\tau_{(i+1)}$, and a new object is detected in τ_i, it is considered that there is a possibility of a handover between τ_i and $\tau_{(i-1)}$, or τ_i and $\tau_{(i+1)}$.

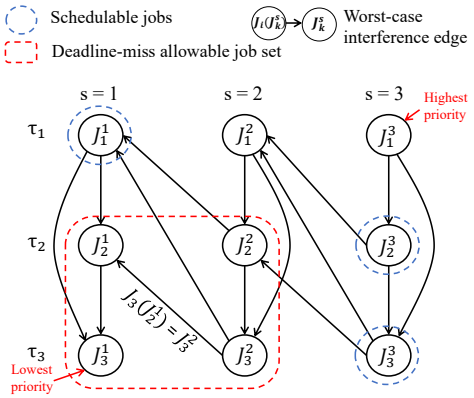


Figure 4: Interference relation for $\tau = \{\tau_1, \tau_2, \tau_3\}$ where τ_1 and τ_3 are the highest and lowest priority task, with m_1=m_2=m_3=2

Global association. If handover identification indicates a handover between two adjacent cameras, the global association can be performed according to the MOT scheduler's decision, which will be detailed in Section 5.3. For two tasks, τ_i and $\tau_{(i-1)}$ with the possibility of a handover, feature extraction is performed on objects from both the current and previous frames (e.g., at times t and t-f) using convolutional neural network (CNN)-based models such as OSNet [19] and IBN-Net [11]. Based on these extracted features, the system matches the two objects exhibiting the highest feature similarity. Once the global association is completed, for the same object that exists in different cameras (e.g., for τ_i and $\tau_{(i-1)}$, which likely have different IDs assigned by τ_i and $\tau_{(i-1)}$ respectively), the object's ID is updated to have a consistent ID across both cameras.

5 MOT SCHEDULER

The thread-level MOT scheduler runs as a background daemon and communicates with the MOT execution pipeline using the shared memory. While the MOT is executing, the scheduler remains in a waiting state and is invoked either when an MOT job is released or completed. The MOT scheduler encompasses the task model, offline stability analysis, and online scheduling algorithm to be represented in the following subsections.

5.1 Task model

We consider an MOT system τ that consists of n MOT tasks. An MOT task $\tau_i \in \tau$ is specified by a tuple of (T_i, C_i) where T_i is a period, and C_i is the worst-case execution time (WCET). A series of jobs are invoked by a task τ_i, each of which has the minimum inter-arrival times for T_i and requires at most C_i time units to be completed. We use m_i to denote the maximum number of allowable consecutive job deadline misses without losing the stability of τ_i. Based on the number of consecutive deadline misses, each job possesses a distinct system state level (SSL), denoted by s. J_i^s is the job of τ_i after missing deadlines s-1 times for $1 \le s \le m_i$+1. From the moment of its release at t, a deadline miss occurs if a job does not complete its execution before or at $t + T_i$. A job is

considered active at t if it has been released at or prior to this time and still has remaining execution at t. We consider fixed-priority (FP) scheduling, in which a job with a higher SSL is given a higher priority as its missed deadline is more likely to compromise stability. Among jobs with the same SSL, a job released from a task with a higher priority (that is pre-defined) is given a higher priority. Figure 4 illustrates the priorities of each job for an example task set $\tau = \{\tau_1, \tau_2, \tau_3\}$ where τ_1 is the highest priority task, τ_3 is the lowest priority task, and all share the same $m_i = 2$. As seen in the figure, each task has three kinds of jobs with distinct SSL values, resulting in a total of nine types of jobs with unique priorities. Specifically, J_1^3 holds the highest priority while J_3^1 is the lowest priority job. We assume that each MOT job executes non-preemptively.

An MOT task τ_i sequentially processes input images from the corresponding camera, which executes continuous detection to determine the location and class of objects and performs association to match objects present in two successive frames. C_i of task τ_i is derived from the sum of the worst-case detection time C_i^D and the worst-case association time C_i^A. As MOT-AS provides different execution options, C_i can vary according to the options. MOT-AS supports two execution options for local associations, namely low-workload local association for $C_i^A(\mathsf{L})$ and high-workload local association for $C_i^A(\mathsf{H})$. This execution can proceed with or without the global association executing for $C_i^A(\mathsf{G})$. As the global association requires DNN-based heavy computation, while the local association does not, we assume that $C_i^A(\mathsf{L}) \le C_i^A(\mathsf{H}) \le C_i^A(\mathsf{G})$ holds. As the computation time for handover identification is negligible, we added it to either $C_i^A(\mathsf{L})$ or $C_i^A(\mathsf{H})$. Therefore, there can be four possible execution times for C_i.

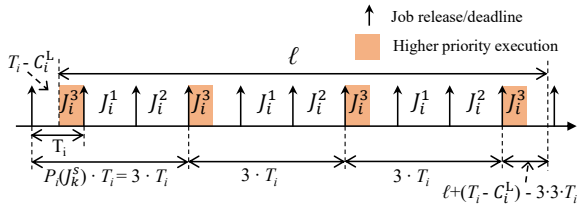
- $C_i^{\mathsf{L}} = C_i^D + C_i^A(\mathsf{L})$,
- $C_i^{\mathsf{H}} = C_i^D + C_i^A(\mathsf{H})$,
- $C_i^{\mathsf{L,G}} = C_i^D + C_i^A(\mathsf{L}) + C_i^A(\mathsf{G})$, and
- $C_i^{\mathsf{H,G}} = C_i^D + C_i^A(\mathsf{H}) + C_i^A(\mathsf{G})$;

therefore $C_i^{\mathsf{L}} \le C_i^{\mathsf{H}} \le C_i^{\mathsf{L,G}} \le C_i^{\mathsf{H,G}}$ holds.

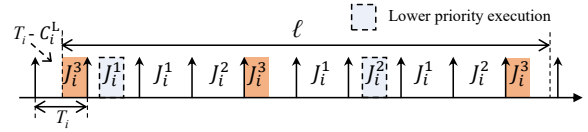
By extending the existing fault-tolerant CPS task model [7] to encompass the multiple MOT execution options aptly, MOT-AS allows up to m_i consecutive deadline misses for τ_i before losing stability. To avoid terminating the running non-preemptive MOT execution at its deadline (i.e., meaning a deadline miss), at every scheduling decision in online scheduling at t, if C_i^{L} for the scheduled job J_i^s at t is strictly larger than J_i^s's earliest deadline from t (i.e., indicating a potential deadline miss), MOT-AS's policy is not to execute J_i^s inducing a deadline miss for J_i^s. We apply a single-processor scheduling model, which allows only a single job to be executed at any given time.

5.2 Offline stability analysis

The offline stability analysis tailored to MOT-AS ensures that every MOT job from the provided task set completes its minimum execution (i.e., C_i^{L} for every τ_i) without more than m_i consecutive deadline misses. To this end, we employ the existing response time analysis (RTA) in which the response time of a job J_k^s is defined as the latest completion time upon its release at a time t. Let R_k^s be the upper bound of the response time of J_k^s. Our new RTA to test the stability of τ derives R_k^s for every job with distinct SSL, and



(a) $W_i(\ell, J_k^s) = 4 \cdot C_i^{\mathsf{L}}$ **case: no execution of lower-priority jobs (i.e.,** J_i^1 **or** J_i^2**)**



(b) $3 \cdot C_i^{\mathsf{L}}$ **case: : some execution of lower-priority jobs (i.e.,** J_i^1 **or** J_i^2**)**

Figure 5: Two scenarios where the execution of τ_i's jobs executing for C_i^{L} with $J_i(J_k^s) = J_i^3$ in an interval of length ℓ is $W_i(\ell, J_k^s) = 4 \cdot C_i^{\mathsf{L}}$ and $3 \cdot C_i^{\mathsf{L}}$

then tests whether R_k^s is less than or equal to T_k to ascertain no deadline miss for J_k^s in the worst-case. If $R_k^s \le T_k$ holds, then J_k^s is called schedulable. Understanding that the response time of J_k^s can be influenced by higher-priority jobs, our initial step is to calculate the maximum execution time of higher-priority jobs from τ_i in an interval of length ℓ. Let $J_i(J_k^s)$ be the lowest-priority job from τ_i with a priority strictly higher than that of J_k^s, and $P_i(J_k^s)$ be the SSL of $J_i(J_k^s)$. If there are no jobs from τ_i with a higher priority than J_k^s, then $J_i(J_k^s) = \emptyset$ and $P_i(J_k^s)$ are not defined. For example, in Figure 4, $J_3(J_2^1) = J_3^2$ and $P_3(J_2^1) = 2$ hold as J_3^2 is the lowest-priority job from τ_3 with a priority strictly higher than that of J_2^1. Subsequently, we denote $W_i(\ell, J_k^s)$ as the upper-bounded execution time of τ_i's jobs with priority strictly higher than J_k^s in an interval of length ℓ.

Figure 5(a) presents the worst-case scenario in which $W_i(\ell, J_k^s)$ occurs for $J_i(J_k^s) = J_i^3$. As seen in Figure 5(a), the leftmost job J_i^3, having a higher priority than J_k^s, initiates its execution at the start of the interval ℓ and completes by the deadline of J_i^3. Because J_i^3 completes either before or at its deadline, the SSL for the subsequent job is set to one, leading to the execution of J_i^1. Subsequently, J_i^3 appears as early as possible, after consecutive deadline misses by J_i^1 and J_i^2, thereby resulting in $W_i(\ell, J_k^s) = 4 \cdot C_i^{\mathsf{L}}$. Note that, based on the scheduling policy of MOT-AS, an MOT job at t, whose WCET is greater than its deadline and does not start its execution. Therefore, J_i^1 and J_i^2 do not perform any execution with deadline misses due to $J_i(J_k^s) = J_i^3$ as seen in Figure 5(a).

Figure 5(b) presents a different situation compared to the worst-case scenario depicted in Figure 5(a) for $W_i(\ell, J_k^s)$. Unlike 5(a), both J_i^1 and J_i^2 have successfully executed once without missing their deadlines between executions of J_i^3. However, J_i^1 and J_i^2 do not interfere with J_k^s because of a lower priority than J_k^s but delay the appearance of J_i^3. As a result, J_i^3 executes three times for C_i^{L} as shown in Figure 5(b), while J_i^3 executes four times in as shown Figure 5(a) for $W_i(\ell, J_k^s)$.

Let us define a job of $J_i(J_k^s) \neq \emptyset$ in the interval $[t, t+\ell)$ as a carry-in job if its release is earlier than t (e.g., the leftmost job of J_i^3 in Figure 5(a)). Also, a job is referred to as a body job if its release is the same as or later than t and its deadline is the same as or earlier than $t+\ell$ (e.g., two middle jobs of J_i^3 in Figure 5(a)). Additionally, let us refer to a job as a carry-out job if its release is earlier than $t+\ell$ and its deadline is later than $t+\ell$ (e.g., the rightmost job of J_i^3 in Figure 5(a)). For a job J_k^s of a task τ_k with system state level s, the minimum execution C_i^{L} of $J_i(J_k^s) \neq \emptyset$ in an interval of length ℓ is upper-bounded by

$$W_i(\ell, J_k^s) = n_i(\ell, J_k^s) \cdot C_i^{\mathsf{L}} + min(\ell + (T_i - C_i^{\mathsf{L}}) - n_i(\ell, J_k^\ell) \cdot P_i(J_k^\ell) \cdot T_i, C_i^{\mathsf{L}}), \quad (1)$$

where $n_i(\ell, J_k^s)$ is the number of carry-in or body jobs of τ_i of which execution is fully performed in an interval of length ℓ, which is obtained by

$$n_i(\ell, J_k^s) = \left\lfloor \frac{\ell + (T_i - C_i)}{P_i(J_k^s) \cdot T_i} \right\rfloor. \quad (2)$$

The first term in Equation (1) represents the maximum execution from carry-in or body jobs of τ_i, and the second term is from a carry-out job.

Lemma 1. *For a job J_k^s of a task τ_k with system state level s, $W_i(\ell, J_k^s)$ in Equation (1) upper-bounds the execution of $J_i(J_k^s) \neq \emptyset$ in an interval of length ℓ.*

Proof. Let us assume the worst-case scenario where the execution of $J_i(J_k^s)$ is maximized in an interval $[t, t+\ell)$. In this scenario, the carry-in job begins its execution at time t and is completed by its deadline. For the body-job execution to start as early as possible, jobs from τ_i with a priority lower than $J_i(J_k^s)$ (if any) must miss their deadlines. Then, the body and carry-out jobs start their execution at their release upon its release. For this execution configuration of the carry-in, body, and carry-out jobs, if we shift the beginning of the interval (e.g. at t) to the left (e.g., to $t^- < t$), the execution of the carry-out job can decrease within that interval. Conversely, shifting to the right (e.g., to $t^+ > t$) could reduce the carry-in job's execution in the interval. Both cases contradict the assumption, and thus the lemma holds. □

For a given job J_k^s and task τ_i, if there's no job $J_i(J_k^s)$ (i.e., $J_i(J_k^s) = \emptyset$), we set $W_i(\ell, J_k^s)$ to zero. Based on Lemma 1, we can iteratively calculate an upper-bound of the response time R_k^s of J_k^s with minimum execution for C_k^{L}, which is similar to RTA under non-preemptive FP scheduling [4].

Theorem 1. *Consider a task set τ scheduled by non-preemptive FP. The upper-bounded response time R_k^s of J_k^s with minimum execution for C_k^{L} is derived by R^x of the following formula such that it satisfies the relation $R_k^{x+1} \leq R_k^x$, initialized with $R_k^0 = C_k^{\mathsf{L}}$:*

$$R^{x+1} \leftarrow C_k^{\mathsf{L}} + \max_{\tau_j \in LP(\tau_k)} C_j^{\mathsf{L}} + \sum_{\tau_i \in \tau - \{\tau_k\}} W_i(R^x, J_k^s). \quad (3)$$

Proof. To ensure J_k^s responds within the interval R^x, three types of execution must be met within that interval: from (i) higher priority job(s), (ii) an executing lower-priority job (that incurs blocking), and (iii) a job of J_k^s itself. Based on Lemma 1, (i) cannot be larger than $\sum_{\tau_i \in \tau - \{\tau_k\}} W_i(R^x, J_k^s)$. Also, due to non-preemptive characteristics, J_k^s can be blocked by (ii), which does not exceed $\max_{\tau_j \in \mathsf{LP}(\tau_k)} C_j^{\mathsf{L}}$. Lastly, (iii) is upper-bounded by the first term, C_k^{L}. Thus, the theorem holds. □

Using Theorem 1, we determine that J_k^s is schedulable if R_k^s is less than T_k. Stability compromised by τ_k necessitates m_i+1 consecutive deadline misses. Hence, if every task τ_k has at least one schedulable job J_k^s among m_i+1 jobs with different SSL, stability is ensured.

We employ Theorem 1's RTA to conduct our stability analysis for τ. This process starts from a job with the lowest SSL of the highest-priority task and progresses to a job with the highest SSL of the lowest-priority task, examining the schedulability of each J_k^s. It is essential to note that if J_k^s is found to be schedulable, job J_k^{s+1} does not exist. Thus, during the sequential application of Theorem 1's RTA, any job J_i deemed non-existent exerts no interference on J_k^s during its RTA.

For a task set τ that clears the stability analysis, we introduce the *deadline-miss allowable job set*, denoted by J^{M}. This set comprises the union of jobs (if any) with an SSL lower than the schedulable job J_i^s of each task τ_i. A notable attribute of a job $J_i^s \in J^{\mathsf{M}}$ is that, despite a potential deadline miss, a job with a superior SSL is guaranteed to remain schedulable, ensuring that stability remains guaranteed. As illustrated in Figure 4, if J_1^1, J_2^3, and J_3^3 are deemed schedulable (represented by blue dotted circles), jobs with an SSL below these are categorized as the deadline-miss allowable job set (highlighted within the red dotted lines in the figure).

5.3 Online scheduling algorithm

We now develop NPFP$^{\mathsf{ML}}$ as our online scheduling algorithm. At each scheduling decision at t, NPFP$^{\mathsf{ML}}$ evaluates the feasibility of execution options from MOT-AS excluding C_i^{L} whose stability is guaranteed by the offline stability analysis discussed in Section 5.2. At every scheduling decision at t, the highest-priority active job J_k^s in the ready queue is chosen for scheduling. Let $d_k^{(t)}$ represent the earliest deadline of J_k^s from t. Furthermore, we introduce a function, $\Gamma(J^{\mathsf{M}}, t, t+x)$, which returns `true` if all active jobs—except for the highest priority at t—and upcoming jobs that will be released within the interval $[t, t+x)$ are members of J^{M}; otherwise, it returns `false`. MOT-AS supports the following scenarios, which are also illustrated in Figure 3.

- **S1.** ($t + C_k^{\mathsf{L}} \leq d_k^{(t)} < t + C_k^{\mathsf{H}}$) or ($t + C_k^{\mathsf{H}} \leq d_k^{(t)}$ and $\Gamma(J^{\mathsf{M}}, t, t + C_k^{\mathsf{H}})$=`false`): NPFP$^{\mathsf{ML}}$ conducts detection and local association for C_k^{L}, with inherent stability ensured by the offline analysis for all MOT executions for C_k^{L}.
- **S2.** $t + C_k^{\mathsf{H}} \leq d_k^{(t)} < t + C_k^{\mathsf{L,G}}$ and $\Gamma(J^{\mathsf{M}}, t, t + C_k^{\mathsf{H}})$=`true`: NPFP$^{\mathsf{ML}}$ performs detection and local association for C_k^{H}, preserving stability since any potential deadline miss of a job J_i^s in J^{M} ensures a higher SSL job remains schedulable.
- **S3.** $t+C_k^{\mathsf{L,G}} \leq d_k^{(t)} < t+C_k^{\mathsf{H,G}}$ and $\Gamma(J^{\mathsf{M}}, t, t+C_k^{\mathsf{L,G}})$=`true`: detection and local association for C_k^{L} are executed, followed by a global association for $C_k^A(\mathsf{G})$ in case of a handover.

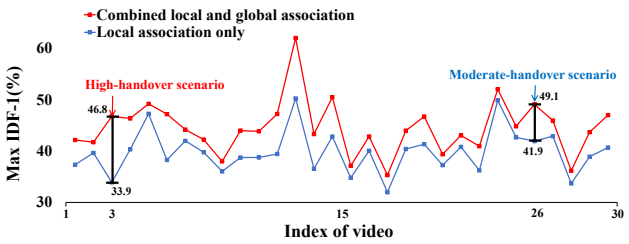


Figure 6: Comparison of maximum accuracy achieved with combined local and global association versus local association only on Tesal V100

Environment	Time (ms)	C_i^D	C_i^A			C_i^{sche}
			$C_i^A(\mathsf{L})$	$C_i^A(\mathsf{H})$	$C_i^A(\mathsf{G})$	
Tesla V100	Average	12.2	1.2	3.3	21.7	0.1
	Maximum	23.3	5.8	16.9	65.3	0.2
Jetson Xavier	Average	39.7	3.4	10.0	51.2	0.2
	Maximum	42.0	15.2	43.5	185.2	0.7

Table 1: Execution time measurement

- **S4.** $t + C_k^{\mathsf{H,G}} \leq d_k^{(t)}$ and $\Gamma(J^{\mathsf{M}}, t, t + C_k^{\mathsf{H,G}})$=true: detection and local association for C_k^{H} are executed, and in the event of a handover, a global association for $C_k^A(\mathsf{G})$ is conducted consecutively.
- **S5.** $t + C_k^{\mathsf{L}} > d_k^{(t)}$: the execution for J_k^s is skipped to avoid potential termination during its non-preemptive run, but even if this leads to a deadline miss, stability remains assured due to the offline stability analysis.

Upon the completion of the execution of the highest-priority job (or no execution satisfying the condition in S5), the job is then removed from the ready queue.

6 EVALUATION

6.1 Experimental setup

Hardware and software. To evaluate the performance of MOT-AS, we utilized both a high-end GPU environment and a GPU-enabled embedded board (NVIDIA Jetson Xavier system-on-chip (SoC) [9]). Our first setup (denoted by Tesla V100) features Intel(R) Xeon(R) Silver 4215R CPUs @ 3.20GHz, 251.5GB RAM, and an NVIDIA Tesla V100 GPU. Conversely, our second setup (denoted by Jetson Xavier) boasts a 64-bit 8-core CPU, 32GB RAM, and a 512-core Volta GPU. MOT-AS is developed in Python, incorporating YOLOv5 [17] for a detector, IBN-Net [11] for feature extraction, and OC-SORT [3] for tracker.

Dataset. To evaluate the performance of MOT-AS, we conducted experiments using the Nuscenes dataset [2], which is widely employed for assessing handover-aware multi-object tracking systems. The Nuscenes dataset encompasses six cameras (front, front-right, front-left, rear, rear-right, and rear-left), each containing approximately 200 frames, amounting to around 1,200 frames per video sequence. Among 900 videos, the detector and feature extractor were trained using 750 videos, and the remaining 150 videos were used for testing. Figure 6 illustrates the achieved maximum accuracy when executing both local and global associations for every frame, as opposed to conducting the local association only across 30 videos (among 150 testing videos), under the deployed detector (i.e., YOLOv5) and tracker (i.e., OC-SORT) on Tesla V100. The changes in performance improvements seen during global association are due to the different number of handovers in each video. For instance, in a high-handover scenario (e.g., video index 3), the maximum accuracy when performing global association is 46.8, compared to 33.9 without it, showing a difference of 13.1. In contrast, in a moderate-handover scenario (e.g., video index 26), the accuracy figures are 49.1 and 41.9, reflecting a difference of 7.2. Experiments conducted on the Jetson Xavier exhibit a similar trend. We examined the influence of MOT-AS on accuracy enhancement in relation to these varying handover frequencies on two different platforms, Tesla V100 (in Figure 7) and Jetson Xavier (in Figure 8).

Excution time profiling and scheduling overhead. For our experimental setup, we measured the WCET of MOT-AS's detection (C_i^D) and association (C_i^A) components. The association is categorized into three distinct options within MOT-AS: low-workload $C_i^A(\mathsf{L})$, high-workload $C_i^A(\mathsf{H})$, and global $C_i^A(\mathsf{G})$. Moreover, we assessed the WCET required by the online scheduling algorithm to conduct a scheduling decision (C_i^{sche}). After conducting about 1,000 individual measurements for each component, we derived both the average and max execution times. To ensure consistency in our experiments across different computing environments, we tested the components on both the Jetson Xavier and Tesla V100. Notably, due to the Jetson Xavier's relatively constrained computing capabilities compared to the Tesla V100, it consistently recorded longer execution times. For example, $C_i^A(H)$ peaked at 16.9ms on Tesla V100, in contrast to 43.5ms on Jetson Xavier as shown in Table 1.

Considered approaches. We focus on MOT systems wherein stability cannot be ensured by Theorem 1 for every MOT execution involving both high-workload local association and global association (i.e., $C_i^{\mathsf{H,G}}$), although stability with the minimum execution (i.e., C_i^{L}) is guaranteed. Following this, we explore the following approaches, all of which utilize non-preemptive rate-monotonic (RM) scheduling:

- Local-Only: for each MOT execution, both detection and local association (without global association) are performed, with the latter operating at its maximum workload (i.e., with the minimum confidence threshold) that meets the stability test in Theorem 1.
- Global-Mandate: for each MOT execution, detection, local association, and global association are performed, with the local association operating at its maximum workload (i.e., with the minimum confidence threshold) that meets the stability test in Theorem 1.
- Best-Effort: it first substitutes $d_k^{(t)}$ in S1–S5 in Section 5.3 to the nearest deadline or future release from t and executes based on NPFP$^{\mathsf{ML}}$'s decision only if a single task is active at t; otherwise, it defaults to the minimum execution C_i^{L} (for S1-S4) or no execution (for S5).
- NPFP$^{\mathsf{ML}}$: the online scheduling proposed in Section 5.3.

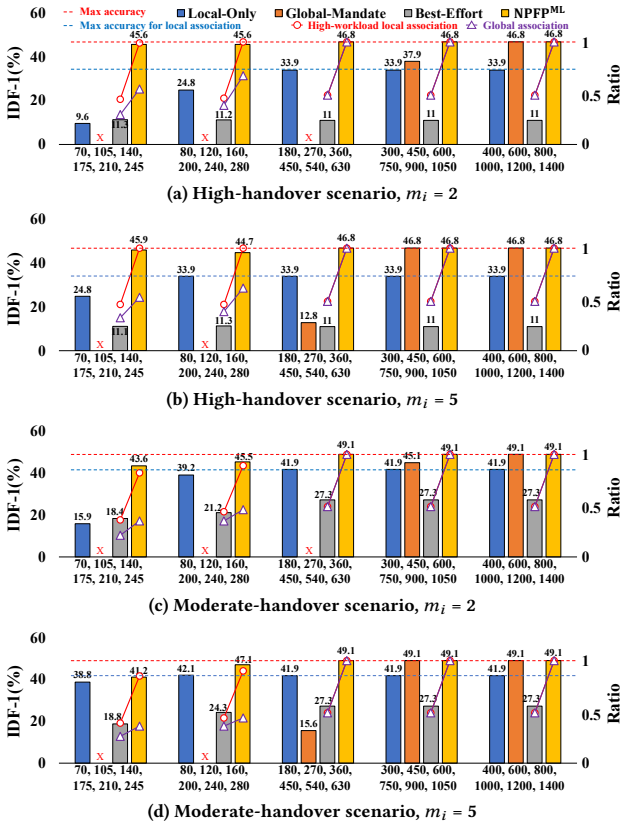


Figure 7: Experiment results on Tesla V100

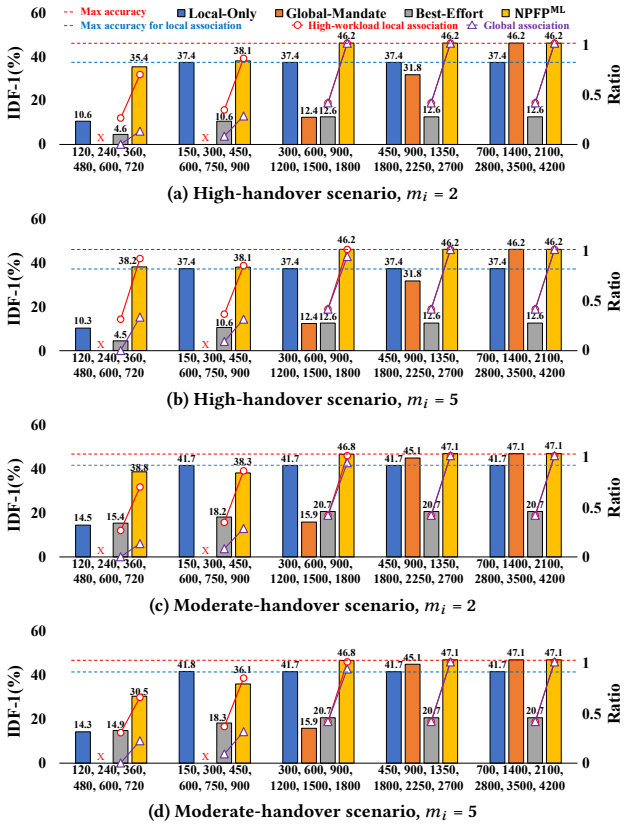


Figure 8: Experiment results on Jetson Xavier

Note that Local-Only represents traditional MOT techniques without considering handover, whereas Global-Mandate takes handover into account. Both have been adapted to meet the stability constraints outlined in the stability analysis proposed by Theorem 1.

6.2 Experiment result

To evaluate the accuracy of multi-camera multi-object tracking, we utilized the IDF−1 metric [16], which captures handovers. We consider various videos presenting different handover scenarios, allowable maximum deadline miss counts m_i, and task periods to demonstrate the efficacy of MOT-AS in enhancing the accuracy of different parameters for handover-aware MOT systems.

Figure 7 presents the performance comparison results executed on the Tesla V100. Frames from a total of six cameras are fed at different intervals, and the period for each MOT task is specified on the x-axis of Figure 7. For the datasets, we consider two types of videos corresponding to high-handover and moderate-handover scenarios, as discussed in Figure 6. The red dashed line and the blue dashed line represent the maximum accuracy achievable when performing both local and global association for every frame, and when performing the local association only, respectively. All six MOT tasks share the same allowable deadline miss counts of two (i.e., m_i = 2 for Figures 7(a) and (c)) and five (i.e., m_i = 5 for Figures 7(b) and (d)). Also, we respectively measured the proportions of high-workload local association executions and global association executions for each approach, during the entire MOT execution. For the Local-Only and Global-Mandate approaches, since the local association operates with only a sole execution option, we did not measure the ratio of high-workload local association executions. Furthermore, the global association is always zero for Local-Only and always one for Global-Mandate. Therefore, we did not separately indicate these proportions for these two approaches in Figures 7 and 8.

As seen in Figure 7(a), for the high-handover scenario with m_i = 2, Local-Only demonstrates improved accuracy as the period of tasks increases, eventually reaching the maximum accuracy achievable solely through local association. For instance, with the first task set, it achieves an accuracy of 9.6. However, from the third task set onwards, it attains its maximum accuracy. On the other hand, Global-Mandate fails the stability test for the first task set and only passes from the fourth task set onwards. As evident from Table 1, this is because the global association demands significantly higher WCET than the high-workload association. Nevertheless, with an accuracy of 37.9 for the fourth task set, Global-Mandate outperforms Local-Only, indicating that while global association requires high WCET, it plays a crucial role in enhancing accuracy. For Best-Effort, due to the constraint that it can operate with an execution option other than the minimum execution (i.e., C_i^{L}) only

when there's a single active job, it shows substantially lower performance across all task sets. For NPFP$^{\text{ML}}$, it exhibits performance close to the max accuracy even with the first task set. This arises because it can perform high-workload local and global associations together, or either one of them at run-time on scenarios S2–S4. As evident from Table 1, there is a significant gap between the average execution time and the maximum execution time for each component. This indicates that the actual execution of the components at run-time finishes much smaller than their WCET, suggesting that there are ample opportunities for online scheduling decisions to opt for high-workload local execution and global association. The ratio for high-workload local execution approaches one from the first task set, while the ratio for global execution starts nearing one from the third task set.

As depicted in Figure 7(b), for $m_i = 5$, both Local-Only and Global-Mandate demonstrate higher accuracy compared to when $m_i = 2$. This enhancement can be attributed to the offline stability analysis, which allows more deadline misses for each task. Consequently, each task can undergo a more demanding workload of local association (i.e., a lower confidence threshold) and still pass the stability test. Thus, Local-Only and Global-Mandate with $m_i = 5$ operate with a more intensive workload, resulting in improved accuracy when compared to their counterparts with $m_i = 2$. Observing Figures 7(c) and (d), it is evident that the moderate-handover scenario presents a trend similar to the high-handover scenario depicted in Figures 7(a) and (b). However, the attainable maximum accuracy varies depending on whether the global association is performed.

Figure 8 presents an evaluation conducted on Jetson Xavier, using the same experimental settings as Figure 7 for the Tesla V100, except for the task period. Table 1 indicates that the Jetson Xavier, an embedded board environment, yields a higher WCET than the Tesla V100, which operates in a high-end GPU environment. Given this observation, we investigated five different task sets with periods greater than those in Figure 7. Similar to the results observed for Tesla V100, each method exhibits analogous trends on Jetson Xavier.

7 CONCLUSION

In this paper, we proposed MOT-AS, a novel handover-aware MOT execution and scheduling framework tailored for AVs with multi-cameras, which aims to maximize tracking accuracy without sacrificing system stability. Recognizing the resource constraints of AVs, MOT-AS decomposes the handover-aware MOT execution into tracking handover objects and normal objects, each along with global and local associations. It selectively performs the global association only when essential and carries out local association with multiple execution options to explore the trade-off between accuracy and stability. Building upon MOT-AS, we developed a new scheduling framework encompassing a new MOT task model, offline stability analysis, and online scheduling algorithm to maximize accuracy without compromising stability. We implemented MOT-AS on both high-end and embedded GPU platforms using the Nuscenes dataset, demonstrating enhanced tracking accuracy and stability over conventional MOT systems, irrespective of their handover considerations.

ACKNOWLEDGMENTS

This work was supported by the National Research Foundation of Korea (NRF) grant funded by the Korea government (MSIT) (NRF-2022R1A4A3018824, NRF-2021R1F1A1059277, NRF-2022R1G1A1003531, IITP-2023-RS-2022-00156360, RS-2023-00248143, RS-2023-00250742). This research was also supported by the MSIT(Ministry of Science and ICT), Korea under the ITRC(Information Technology Research Center) support program(IITP-2023-RS-2023-00259061121821038201O1) supervised by the IITP(Institute for Information & Communications Technology Planning & Evaluation)

REFERENCES

[1] Alex Bewley, Zongyuan Ge, Lionel Ott, Fabio Ramos, and Ben Upcroft. 2016. Simple online and realtime tracking. In *IEEE international conference on image processing (ICIP)*. 3464–3468.
[2] Holger Caesar, Varun Bankiti, Alex H Lang, Sourabh Vora, Venice Erin Liong, Qiang Xu, Anush Krishnan, Yu Pan, Giancarlo Baldan, and Oscar Beijbom. 2020. nuscenes: A multimodal dataset for autonomous driving. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*. 11621–11631.
[3] Jinkun Cao, Xinshuo Weng, Rawal Khirodkar, Jiangmiao Pang, and Kris Kitani. 2022. Observation-centric sort: Rethinking sort for robust multi-object tracking. *arXiv preprint arXiv:2203.14360* (2022).
[4] Donghwa Kang, Seunghoon Lee, Hoon Sung Chwa, Seung-Hwan Bae, Chang Mook Kang, Jinkyu Lee, and Hyeongboo Baek. 2022. RT-MOT: Confidence-Aware Real-Time Scheduling Framework for Multi-Object Tracking Tasks. In *IEEE Real-Time Systems Symposium (RTSS)*. IEEE, 318–330.
[5] Philipp Kohl, Andreas Specker, Arne Schumann, and Jurgen Beyerer. 2020. The mta dataset for multi-target multi-camera pedestrian tracking by weighted distance aggregation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops*. 1042–1043.
[6] Hyoeun Lee, Youngjoon Choi, Taeho Han, and Kanghee Kim. 2022. Probabilistically Guaranteeing End-to-End Latencies in Autonomous Vehicle Computing Systems. *IEEE Trans. Comput.* 71, 12 (2022), 3361–3374.
[7] Jinkyu Lee and Kang G Shin. 2017. Development and use of a new task model for cyber-physical systems: A real-time scheduling perspective. *Journal of Systems and Software* 126 (2017), 45–56.
[8] NVIDIA. 2018. Tesla V100. [Online]. Available: https://www.nvidia.com/en-us/data-center/v100/.
[9] NVIDIA. 2018. Xavier Developer Kit. [Online]. Available: https://www.nvidia.com/en-us/autonomous-machines/embedded-systems/jetson-agx-xavier.
[10] NVIDIA. 2023. Orin Developer Kit. [Online]. Available: https://www.nvidia.com/ko-kr/autonomous-machines/embedded-systems/jetson-orin/.
[11] Xingang Pan, Ping Luo, Jianping Shi, and Xiaoou Tang. 2018. Two at once: Enhancing learning and generalization capacities via ibn-net. In *Proceedings of the European Conference on Computer Vision (ECCV)*. 464–479.
[12] Ziqi Pang, Jie Li, Pavel Tokmakov, Dian Chen, Sergey Zagoruyko, and Yu-Xiong Wang. 2023. Standing Between Past and Future: Spatio-Temporal Modeling for Multi-Camera 3D Multi-Object Tracking. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. 17928–17938.
[13] Yijun Qian, Lijun Yu, Wenhe Liu, and Alexander G Hauptmann. 2020. Electricity: An efficient multi-camera vehicle tracking system for intelligent city. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops*. 588–589.
[14] Joseph Redmon, Santosh Divvala, Ross Girshick, and Ali Farhadi. 2016. You only look once: Unified, real-time object detection. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. 779–788.
[15] Shaoqing Ren, Kaiming He, Ross Girshick, and Jian Sun. 2015. Faster R-CNN: Towards real-time object detection with region proposal networks. In *Advances in Neural Information Processing Systems (NeurIPS)*.
[16] Ergys Ristani, Francesco Solera, Roger Zou, Rita Cucchiara, and Carlo Tomasi. 2016. Performance measures and a data set for multi-target, multi-camera tracking. In *European Conference on Computer Vision (ECCV)*. 17–35.
[17] Ultralytics. 2020. YOLOv5 [Online]. Available: https://github.com/ultralytics/yolov5.
[18] Yifu Zhang, Peize Sun, Yi Jiang, Dongdong Yu, Zehuan Yuan, Ping Luo, Wenyu Liu, and Xinggang Wang. 2021. Bytetrack: Multi-object tracking by associating every detection box. *arXiv preprint arXiv:2110.06864* (2021).
[19] Kaiyang Zhou, Yongxin Yang, Andrea Cavallaro, and Tao Xiang. 2019. Omni-scale feature learning for person re-identification. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. 3702–3712.