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

Jinkyu Lee

Department of Computer Science and Engineering
Sungkyunkwan University (SKKU)

Seminar at SAIT | April 24th, 2023

교수/연구실 소개

- 이진규 Jinkyu Lee
  - 성균관대 소프트웨어학과 부교수
  - University of Michigan 박사후 연구원
  - KAIST 전산학과 학사/석사/박사

- 연구분야
  - **실시간 시스템** (주연구분야)
    - 실시간 작업의 실시간성(마감시간내 완료) 보장
  - 소프트웨어 기술을 활용한 배터리 성능 향상
  - 모바일 컴퓨팅

- 실시간 컴퓨팅 연구실(**RTCL**: Real-Time Computing Lab.)
  - https://rtclskku.github.io/website/

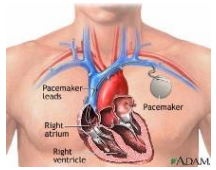
실시간 시스템

- Systems that operate with time constraints (deadlines)
  - Important to produce accurate results before deadlines
  - Usually for safety-critical systems or mission-critical systems

Automotive

Avionics

Medical Devices



Robots

Military

Smartphones

Multimedia

Factory Automation

실시간 시스템

- Systems that operate with time constraints (deadlines)
  - Important to produce accurate results before deadlines
  - Usually for safety-critical systems or mission-critical systems

RTCL 연구 분야 및 성과

Design of *time-predictable* computing systems

+

Timing guarantee and *analysis* of computing systems

Applications

Languages

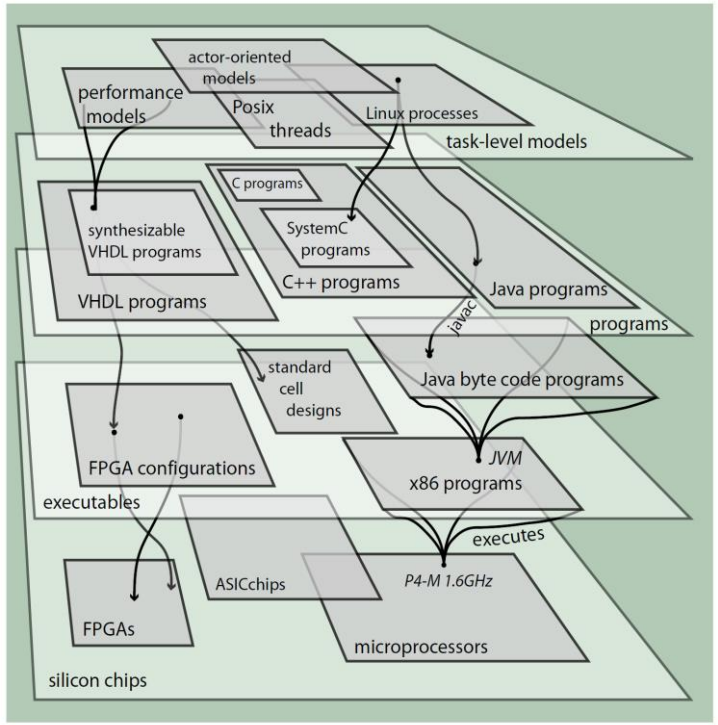
Operating Systems

Middleware

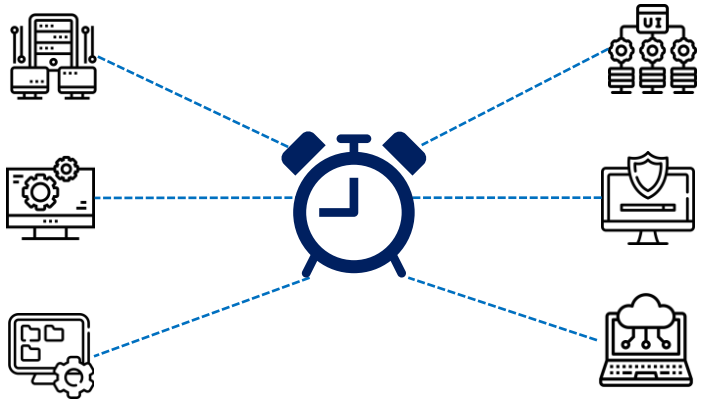
Architecture

Hardware

…



Cyber Physical Systems: Design Challenges, IEEE ISORC, 2008.



RTCL 연구 분야 및 성과

Design of *time-predictable* computing systems

+

Timing guarantee and *analysis* of computing systems

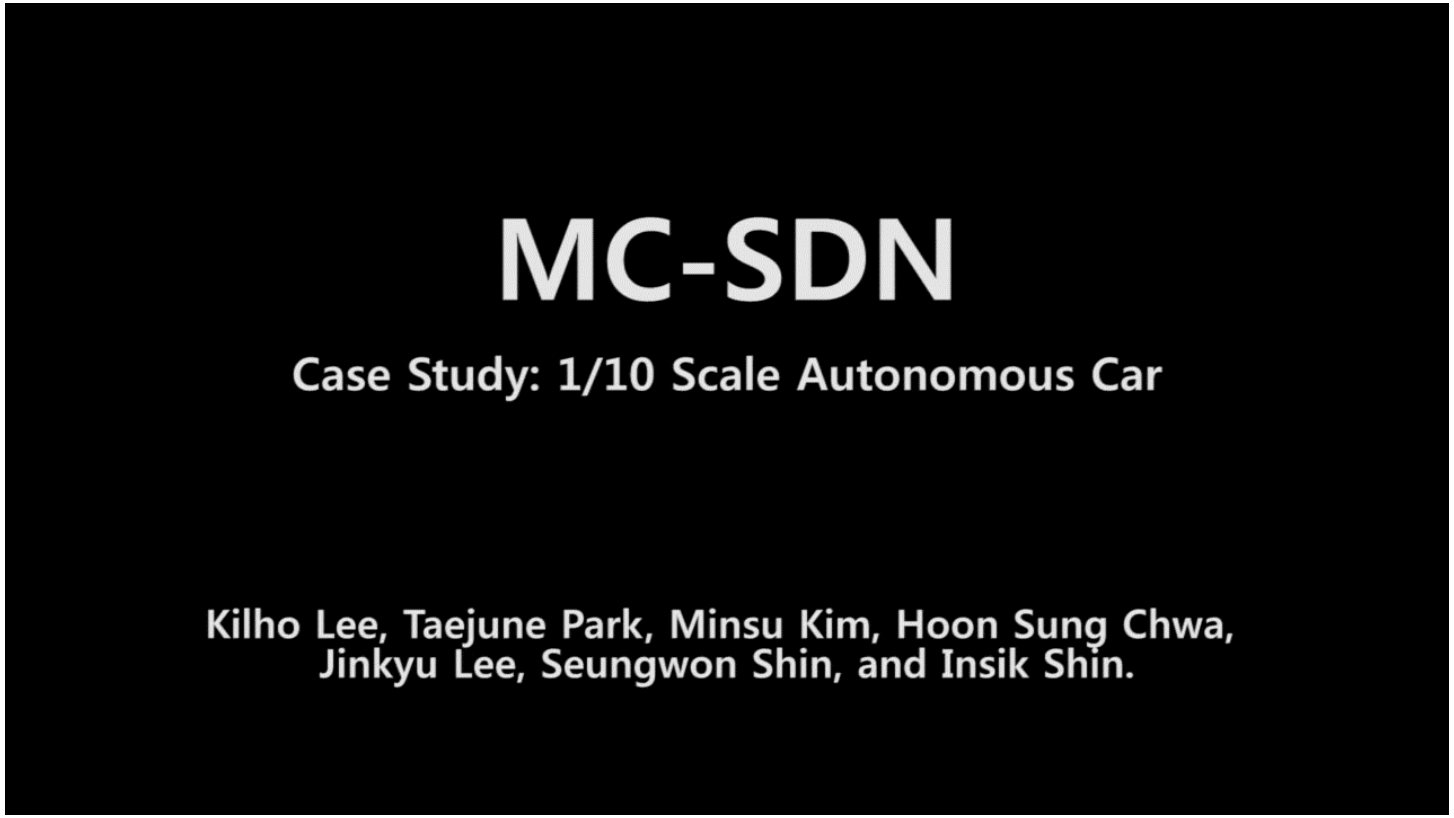
- 마감 시간 내 완료 보장, Time-Predictability 등 타이밍에 관한 요건을 만족해야 하는
  - 컴퓨팅 시스템 디자인 및
  - 이에 대한 타이밍 분석 연구
- 그리고 이와 관련된 노하우를 활용한 연구
  - OS/network 등의 스케줄러, 자원관리, 최적화
  - RT for ML 응용, ML for RT 응용
  - 배터리 성능/파워/에너지 고려한 컴퓨팅 시스템 디자인

RTCL 연구 분야 및 성과

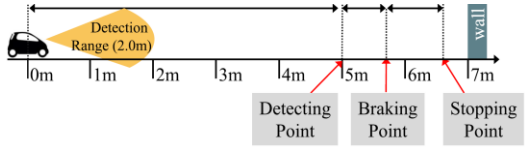
Design of *time-predictable* computing systems

+

Timing guarantee and *analysis* of computing systems



MC-SDN: Supporting Mixed-Criticality Real-Time Communication Using Software-Defined Networking, IEEE IoT Journal, 2019.

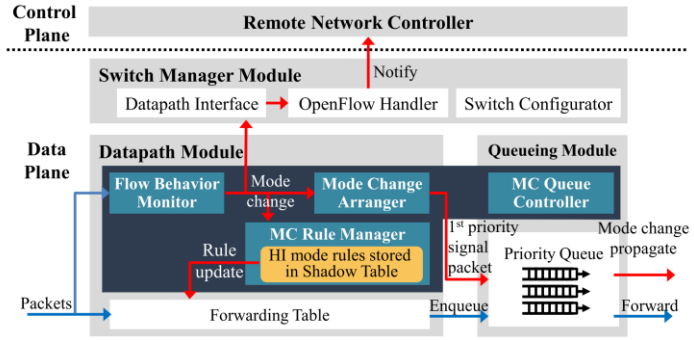


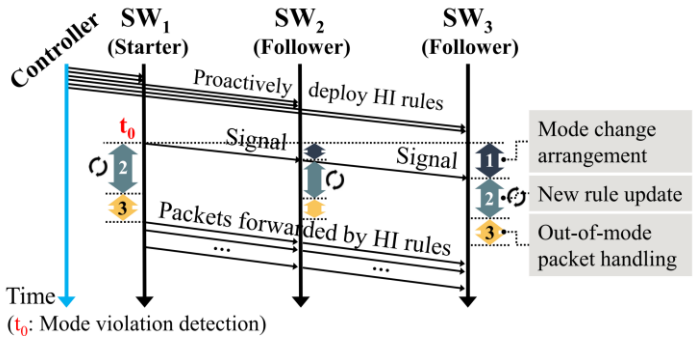
RTCL 연구 분야 및 성과

Design of *time-predictable* computing systems

+

Timing guarantee and *analysis* of computing systems





Total delay for a mode change

$$D_{mc} = D_{arrange} + D_{update} + D_{q-handle}$$

Mode change arrange delay

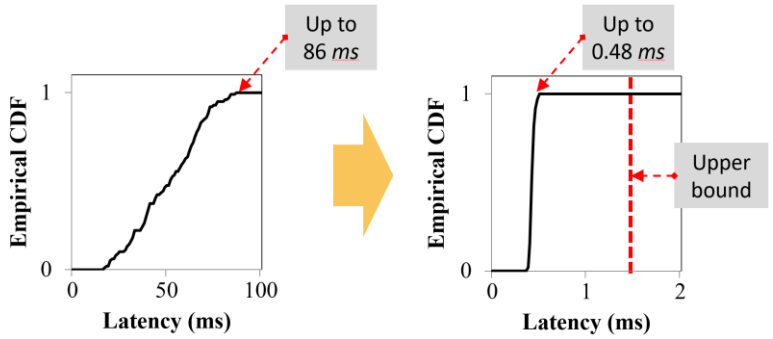
$$D_{arrange} = (d_{trans} + d_{prop} + d_{queue} + d_{proc} + d_{flood}) \cdot N_{link}$$

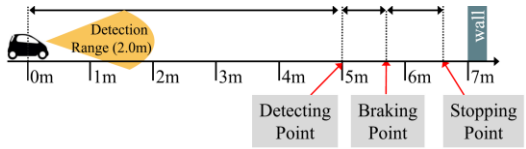
New rule update delay

$$D_{update} = d_{copy} \cdot N_{rule} + d_{u-misc}$$

Out-of-mode packet handling delay

$$D_{q-handle} = d_{q-handle} \cdot N_{packet} + d_{q-misc}$$





MC-SDN: Supporting Mixed-Criticality Real-Time Communication Using Software-Defined Networking, IEEE IoT Journal, 2019.

Jinkyu Lee
Dept. of Computer Science and Engineering

RTCL 연구 분야 및 성과

Design of *time-predictable* computing systems

+

Timing guarantee and *analysis* of computing systems

- 연구 도메인
  - 다양한 컴퓨팅/네트워크 환경
    - CPU: homogeneous/heterogeneous multi-core
    - GPU (Integrated CPU+GPU)
    - Software-defined network, distributed systems
  - 다양한 작업 모델/응용
    - Sequential/parallel/gang task model
    - Mixed-criticality task model
    - DNN, object detection/tracking
  - 다양한 추가 요구 사항
    - Battery-aging minimization
    - Required power supply
    - Peak power minimization

- 연구 성과
  - IEEE **RTSS** (Real-Time Systems Symposium)
    Top1 RT 국제학술대회, 30여편/1년
    - 2022년 단일 년도 3편 논문 발표 **(국내최초)**
    - 최근 11년 연속 논문 발표 2012-2022 **(국내최장)**
    - 총 19편 논문 발표 **(국내최다)**
  - 최근 10년
    - 20+편의 Top-tier 국제학술대회논문
    - 40+편의 SCI 논문

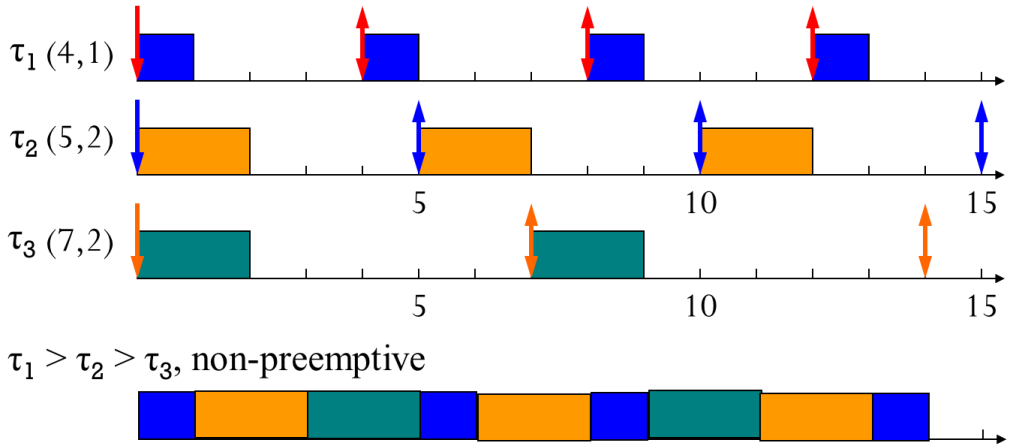
RTCL 연구 분야 및 성과

Design of *time-predictable* computing systems

+

Timing guarantee and *analysis* of computing systems

- Scheduling
  - 어떤 작업을 언제 실행하는지 결정



- Schedulability analysis (for offline timing guarantee)
  - 주어진 scheduling policy 하에서 마감시간내 작업 완료 보장
  - Scheduling policy, 작업 모델, workload, 컴퓨팅 플랫폼에 따라 달라짐

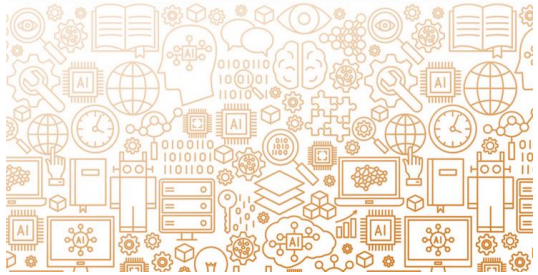
$$\sum_{\tau_i \in \tau} \frac{C_i}{T_i} \leq 1.0$$

$$R_k^0 = C_k \qquad \text{Iterate while } R_k^s > R_k^{s-1}$$

$$R_k^s = C_k + \sum_{\tau_i \in \mathsf{HI}(\tau_k)} \left\lceil \frac{R_k^{s-1}}{T_i} \right\rceil \cdot C_i$$

RTCL 최근 연구 / 관심 주제

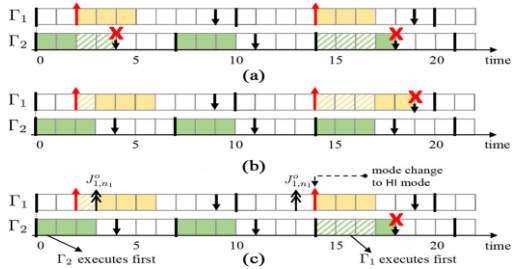
ML



Novel Approach

ML for RT

RT Challenges

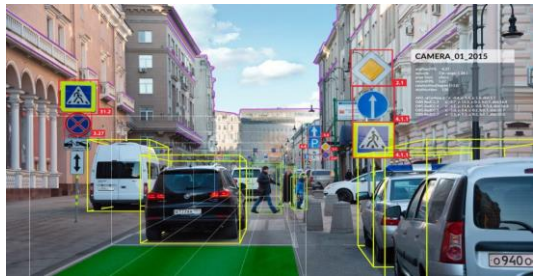


Processing Characteristics

...

Domain Knowledge

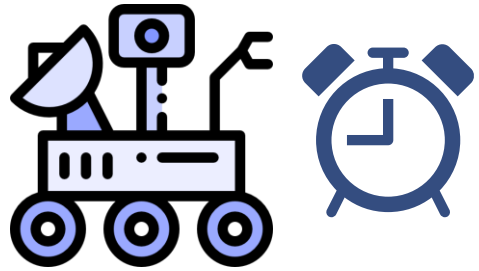
...



ML Applications

RT for ML

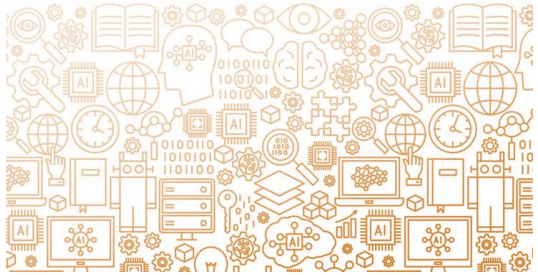
Timing Guarantee



RT

최근 연구 / 관심 주제: ML for RT

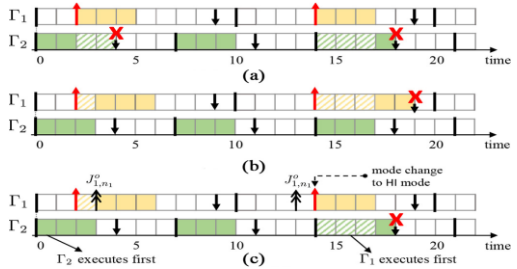
ML



Novel Approach

ML for RT

RT Challenges



- Goal
  - Solving challenging problems in the area of real-time systems
    - By the collaboration of machine learning techniques *and* domain knowledge of real-time systems
- Research trend
  - Problem by problem approach (at least so far)

최근 연구 / 관심 주제: ML for RT

Selecting the "best" priority assignment

$\Theta_{(\tau, P, m)} \leq \Theta$ by heuristic priority assignments (e.g., DMPO, DkC)

Pseudo-premier priority assignments

Priority		Selected by heuristics		System hazard
(τ_1, τ_2, τ_3)	→	(τ_1, τ_2, τ_3)	→	$\Theta = 0.7$
(τ_1, τ_3, τ_2)	→	X	→	$\Theta = 0.5$
(τ_2, τ_1, τ_3)	→	(τ_2, τ_1, τ_3)	→	$\Theta = 1.0$
(τ_2, τ_3, τ_1)	→	X	→	$\Theta = 0.7$
(τ_2, τ_3, τ_1)	→	X	→	$\Theta = 1.0$
(τ_2, τ_3, τ_1)	→	(τ_2, τ_3, τ_1)	→	$\Theta = 0.6$

Priority assignments

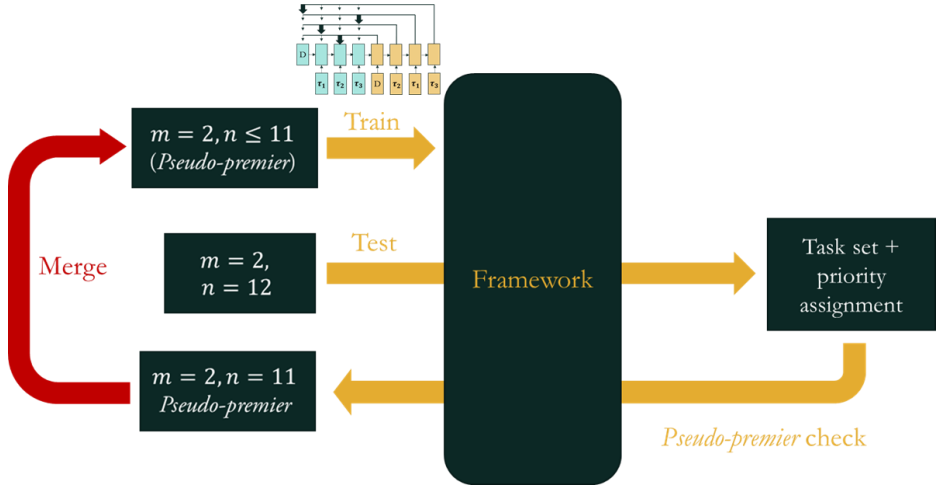
$\tau_1, \tau_2, \tau_3, \tau_4, \tau_5$
$\tau_1, \tau_2, \tau_3, \tau_5, \tau_4$
$\tau_1, \tau_2, \tau_4, \tau_3, \tau_5$
$\tau_1, \tau_2, \tau_4, \tau_5, \tau_3$
$\tau_1, \tau_2, \tau_5, \tau_3, \tau_4$
$\tau_1, \tau_2, \tau_5, \tau_4, \tau_3$
$\tau_5, \tau_4, \tau_3, \tau_1, \tau_2$
$\tau_5, \tau_4, \tau_3, \tau_2, \tau_1$

5! =120 cases

Priority assignments

$\tau_1, \tau_2, \tau_3, \ldots, \tau_{13}, \tau_{14}, \tau_{15}$
$\tau_1, \tau_2, \tau_3, \ldots, \tau_{13}, \tau_{15}$
$\tau_1, \tau_2, \tau_3, \ldots, \tau_{14}, \tau_{13}, \tau_{15}$
$\tau_1, \tau_2, \tau_3, \ldots, \tau_{14}, \tau_{15}, \tau_{13}$
$\tau_1, \tau_2, \tau_3, \ldots, \tau_{15}, \tau_{13}, \tau_{14}$
$\tau_1, \tau_2, \tau_3, \ldots, \tau_{15}, \tau_{14}, \tau_{13}$
$\tau_{15}, \tau_{14}, \tau_{13}, \ldots, \tau_3, \tau_1, \tau_2$
$\tau_{15}, \tau_{14}, \tau_{13}, \ldots, \tau_3, \tau_2, \tau_1$

15! = 1,307,674,368,000 cases

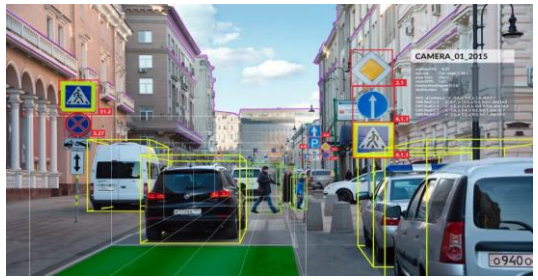


ML for RT: Priority Assignment Using Machine Learning, IEEE **RTAS** 2021

A Graph Attention Network Approach to Partitioned Scheduling in Real-Time Systems, To be submitted

최근 연구 / 관심 주제: RT for ML

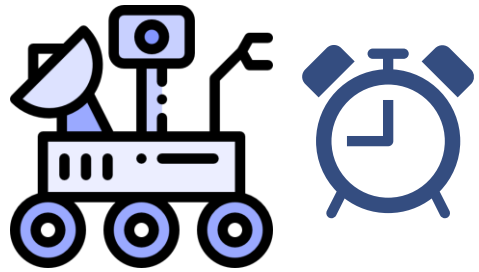
- Goal
  - Design of time-predictable (shorter) execution for DNN tasks
  - Analysis of execution delay upper-bound (timing guarantee)
  - Energy consumption reduction
- Control knob
  - Execution order (job scheduling)
  - Execution behavior (e.g., pipeline execution, CPU/GPU concurrent execution), batch execution
  - Approximation



ML Applications

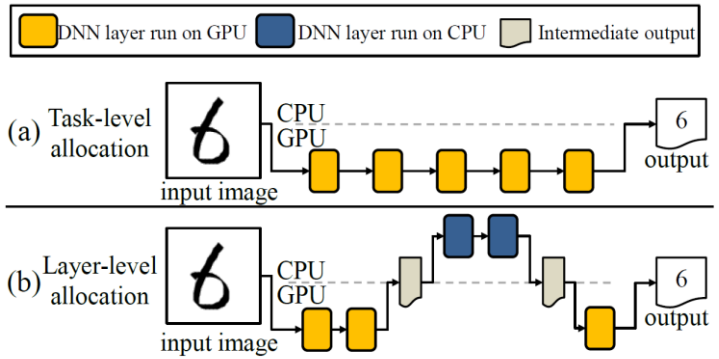
RT for ML

Timing Guarantee

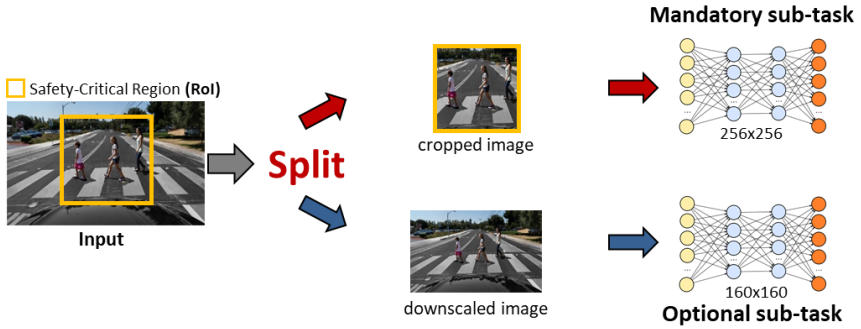


RT

최근 연구 / 관심 주제: RT for ML

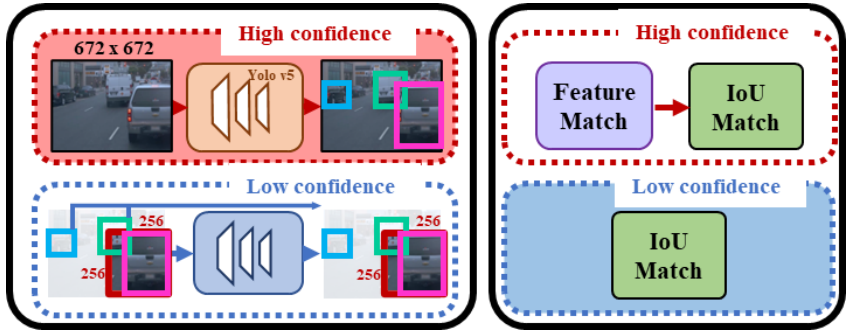


LaLaRAND: Flexible Layer-by-Layer CPU/GPU Scheduling for Real-Time DNN Tasks, IEEE **RTSS** 2021

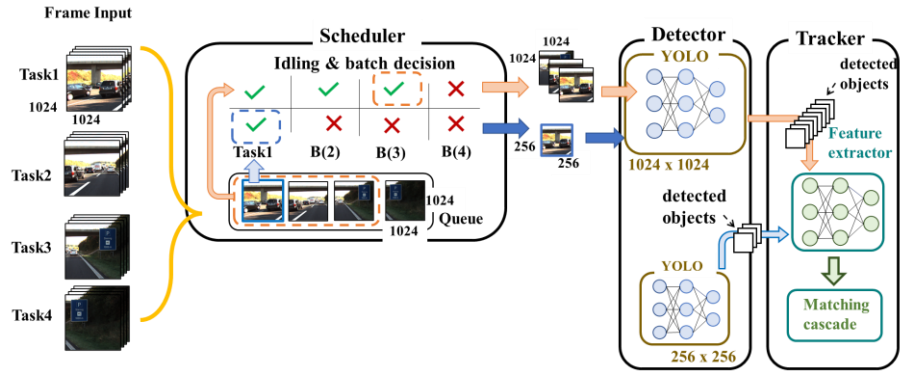


DNN-SAM: Split-and-Merge DNN Execution for Real-Time Object Detection, IEEE **RTAS** 2022

RT-MOT: Confidence-Aware Real-Time Scheduling Framework for Multi-Object Tracking Tasks, IEEE **RTSS** 2022

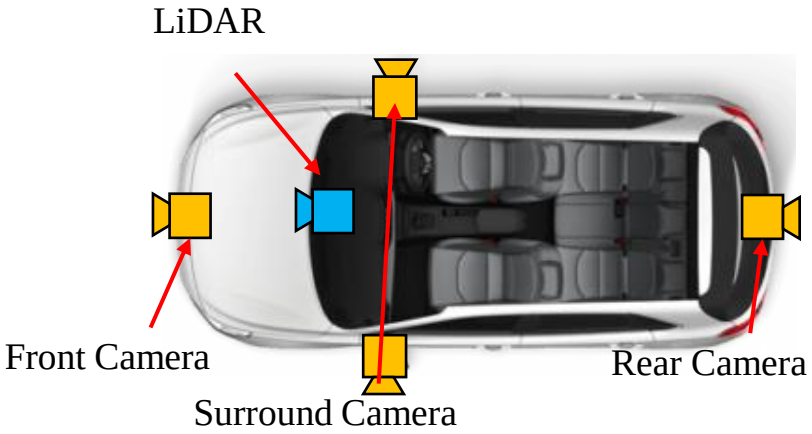


Supporting Batch Execution for Real-Time Multi-Object Tracking Tasks, Submitted to a top-tier conference



Background: Safety-Critical Tasks

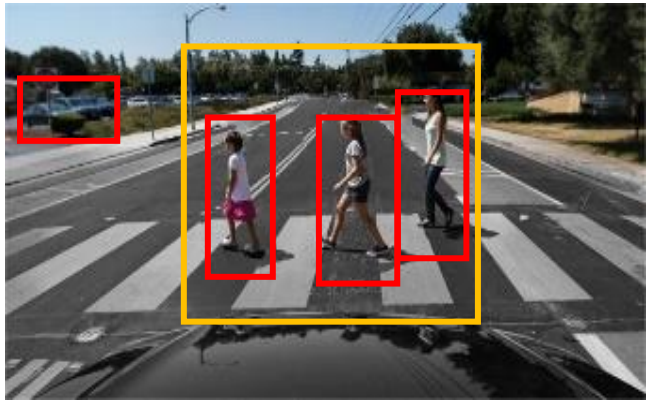
- Autonomous driving
  - Object detection
  - Object tracking
  - Traffic light detection
  - Lane/flow detection
  - Trajectory prediction
  - …



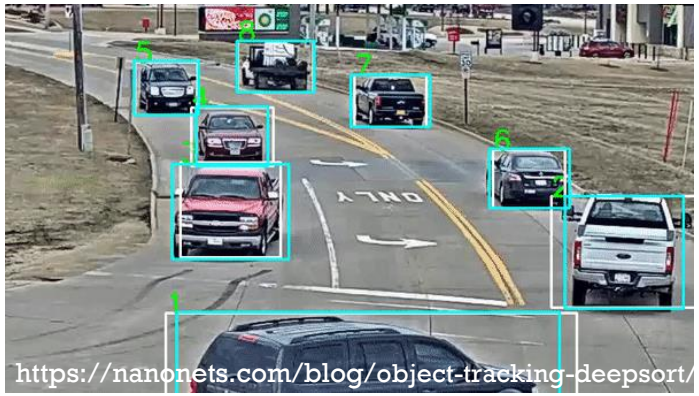
Background: Safety-Critical Tasks

- Autonomous driving

Object Detection

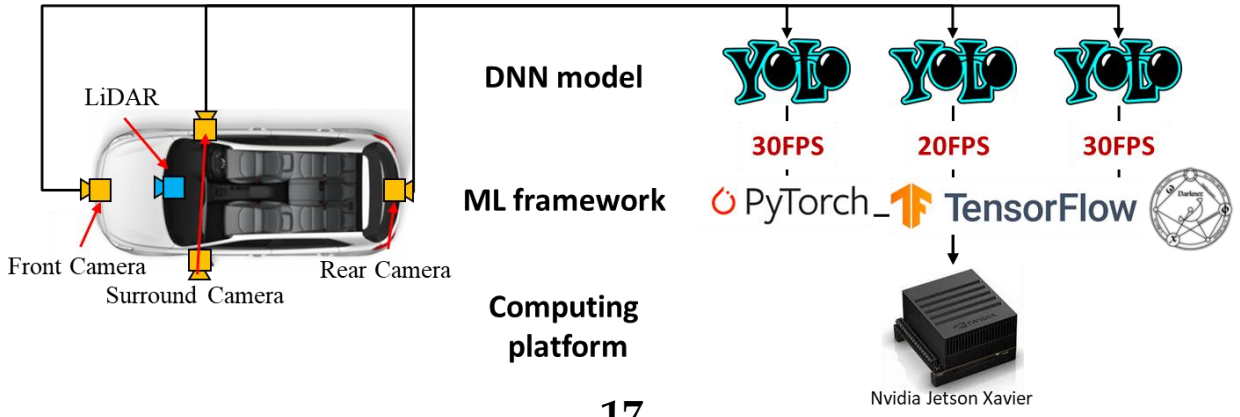


Object Tracking



- Multiple DNN tasks share limited computing resources
- Timely inference with real-time guarantees is essential!

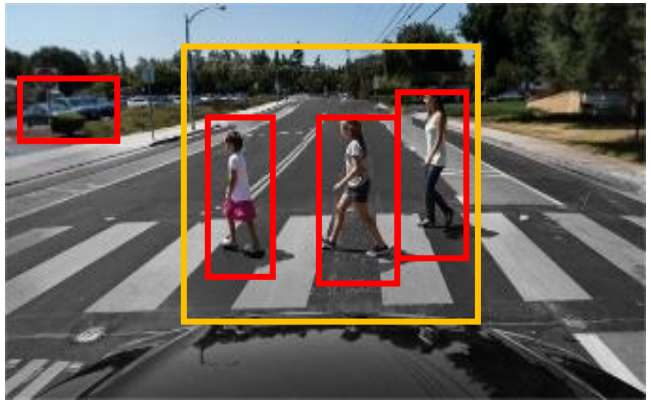
Real-Time Systems



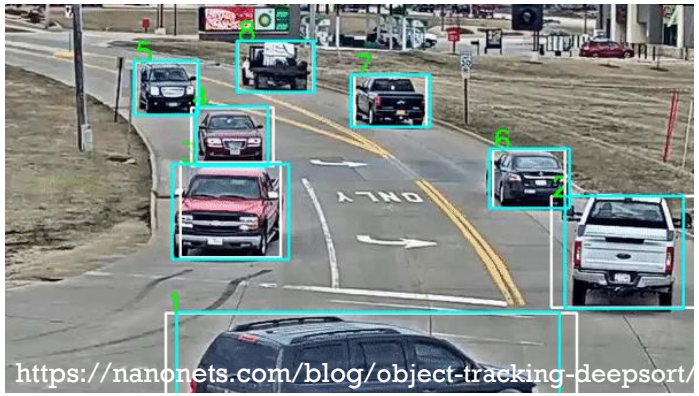
Background: Safety-Critical Tasks

- Autonomous driving

Object Detection



Object Tracking



How can we ensure timely inference of safety-critical tasks?

- on limited computing resources
- while maximizing their accuracy

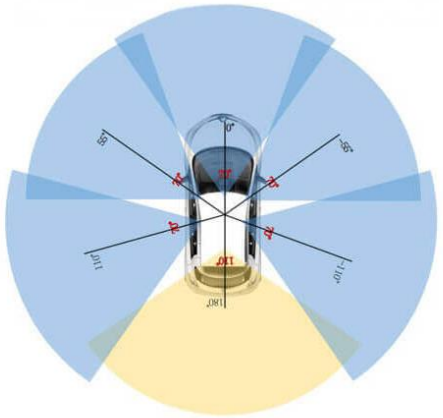
RT-MOT for Object Tracking

- How to achieve *accurate* and *timely* multi-object tracking (MOT)?

RT-MOT: Confidence-Aware Real-Time Scheduling Framework for Multi-Object Tracking Tasks, IEEE **RTSS** 2022

Multi-Object Tracking in Autonomous Vehicle

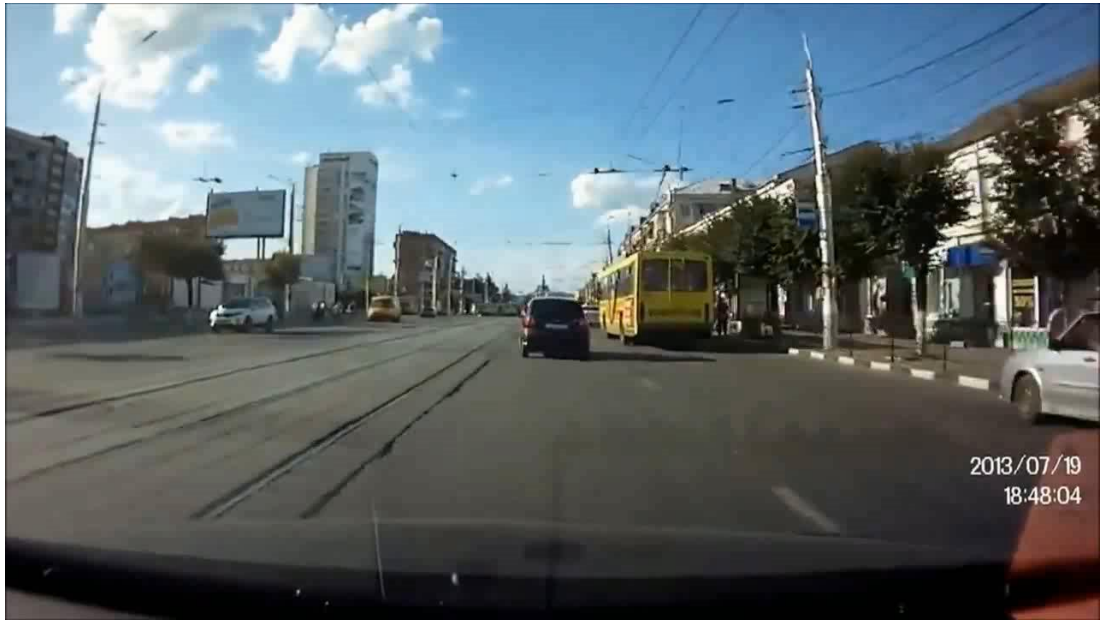
- Autonomous Vehicle (AV) system
  - Concurrent perceptions with multiple sensors (e.g., LiDAR, Cameras)
  - Deep learning-based multi-object tracking (MOT) tasks



Cameras in autonomous vehicle

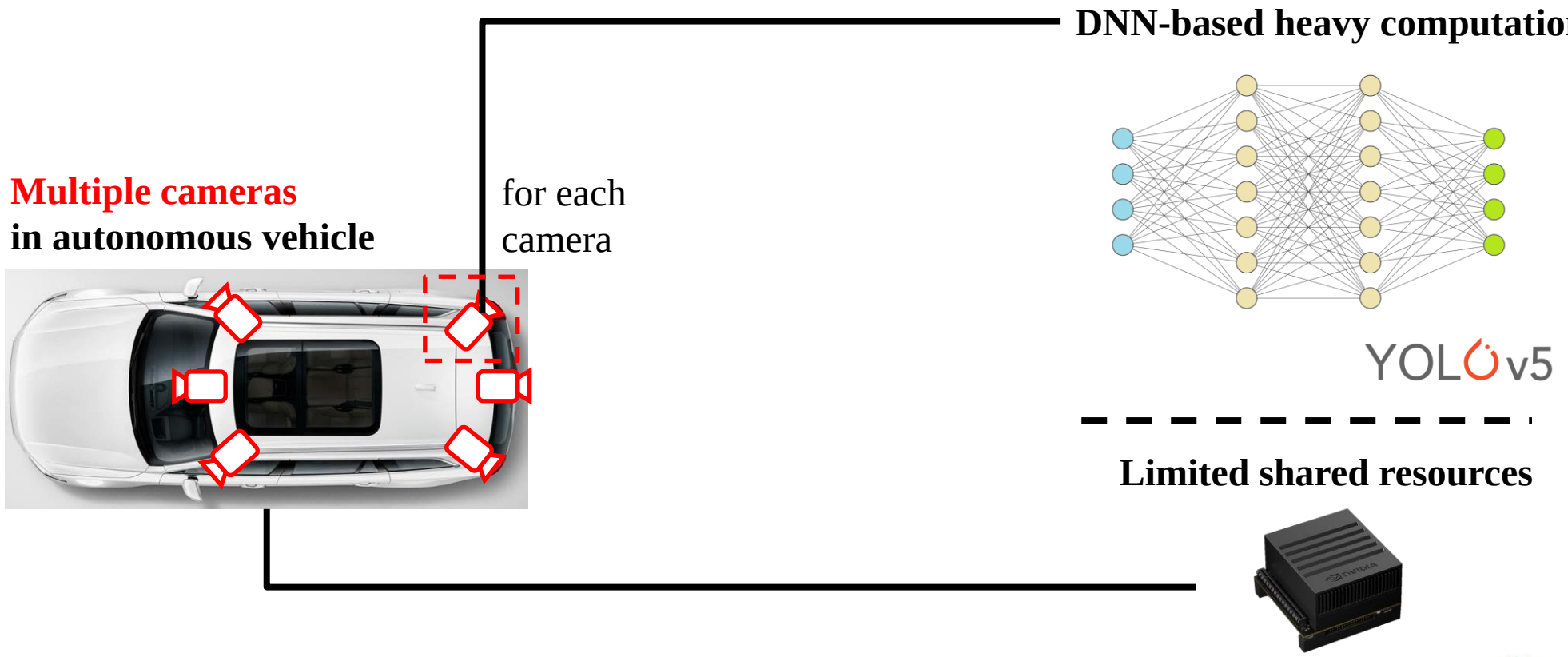
Multi-Object Tracking in Autonomous Vehicle

- For the safety of autonomous vehicles, we aim at satisfing
  - (R1) timing guarantee of all tracking tasks
  - (R2) tracking accuracy maximization



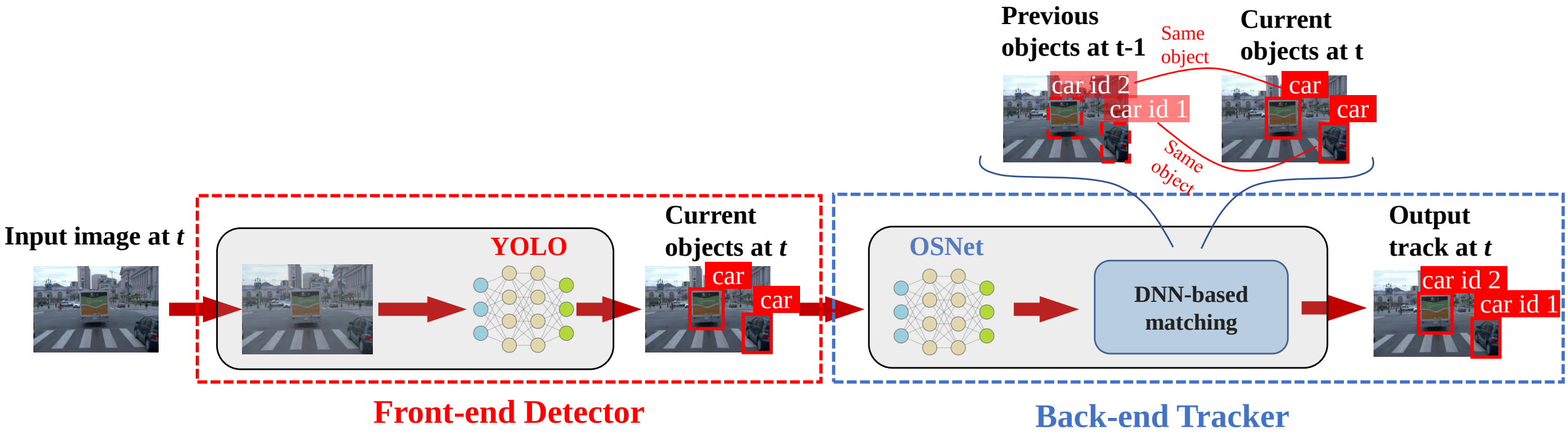
Multi-Object Tracking in Autonomous Vehicle

- Challenging to satisfy both (R1) timing guarantee and (R2) accuracy maximization.
  - Multiple MOT tasks share limited computing resources while requiring heavy computation.



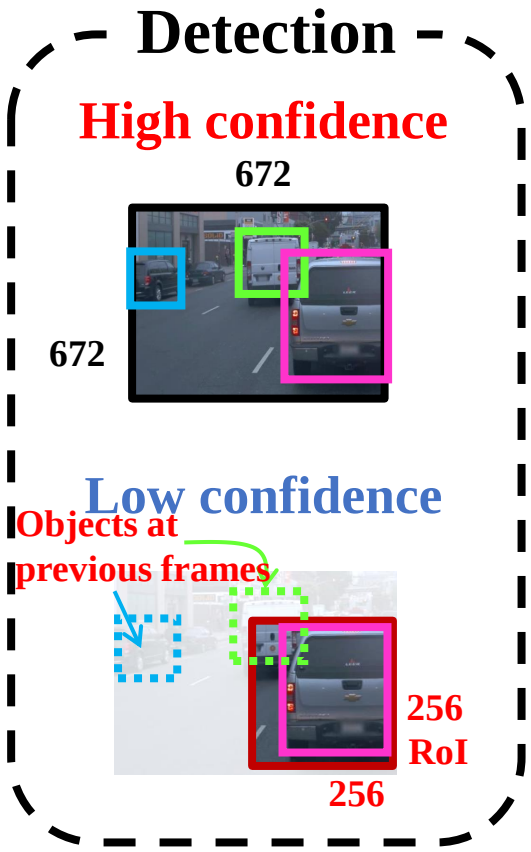
Target multi-object-tracking Architecture

- DNN-based Tracking-by-detection
  - **Front-end detector**: detection of objects in a single video frame
  - **Back-end tracker**: association to match objects detected in the current frame with those from previous frames



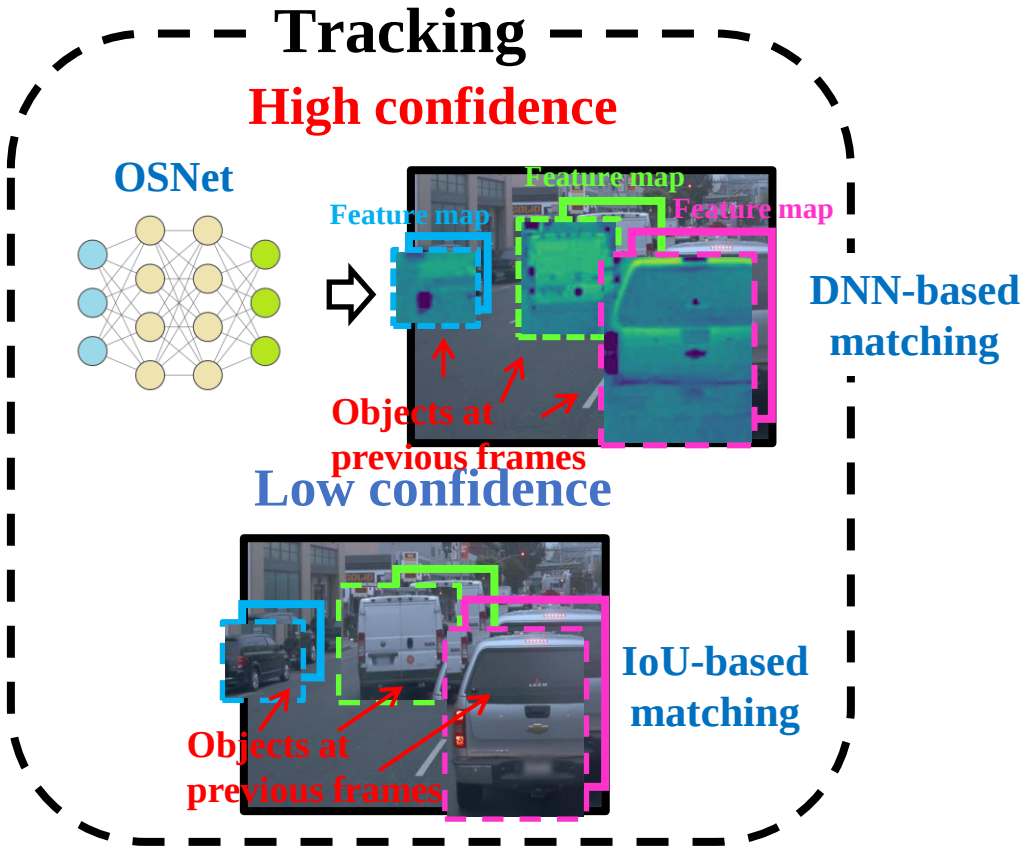
Motivational Observation

- On limited resources
  - Detection can provide a trade-off between R1 (timing) and R2 (accuracy)
    - High-confidence: detect full-frame detection (=**high accuracy, high computation cost**)
    - Low-confidence: detect partial-frame detection (=**low accuracy, low computation cost**)



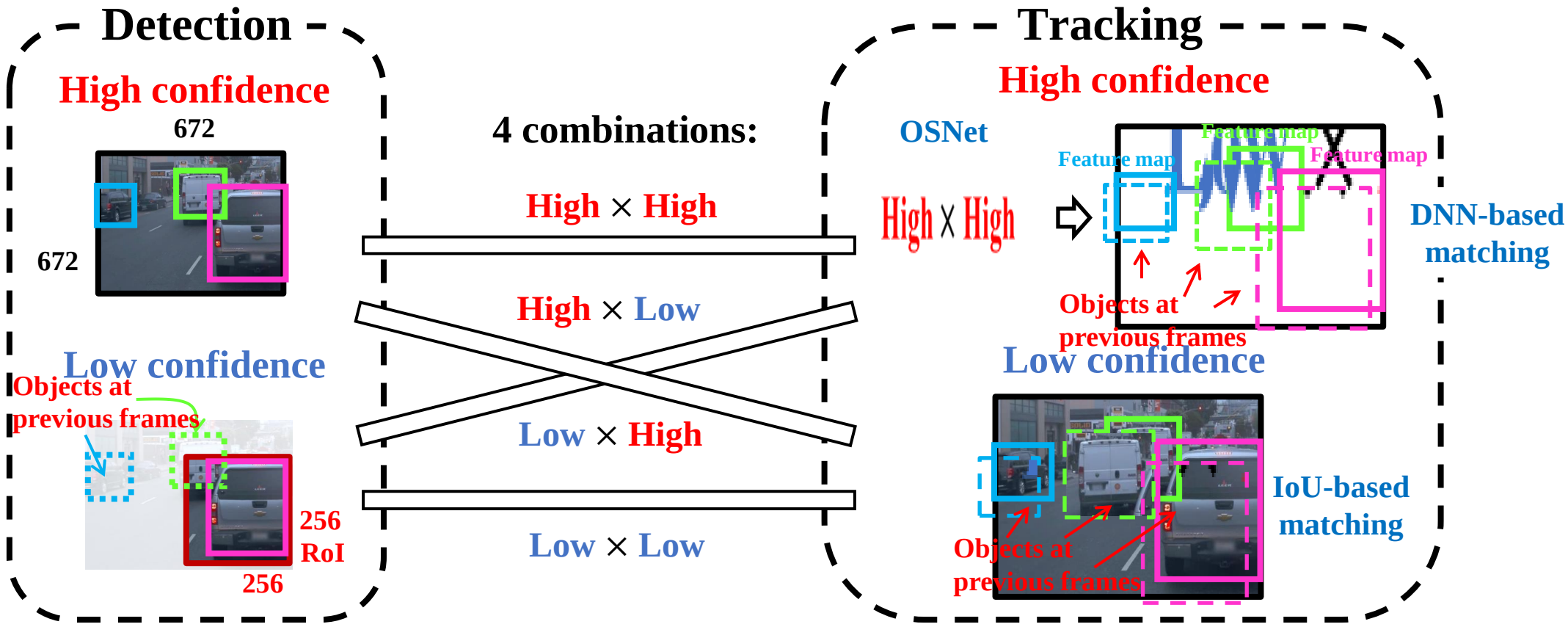
Motivational Observation

- On limited resources
  - Association can provide a trade-off between R1(timing) and R2(accuracy)
    - High-confidence: DNN-based matching (=**high accuracy, high computation cost**)
    - Low-confidence: IoU-based matching (=**low accuracy, low computation cost**)



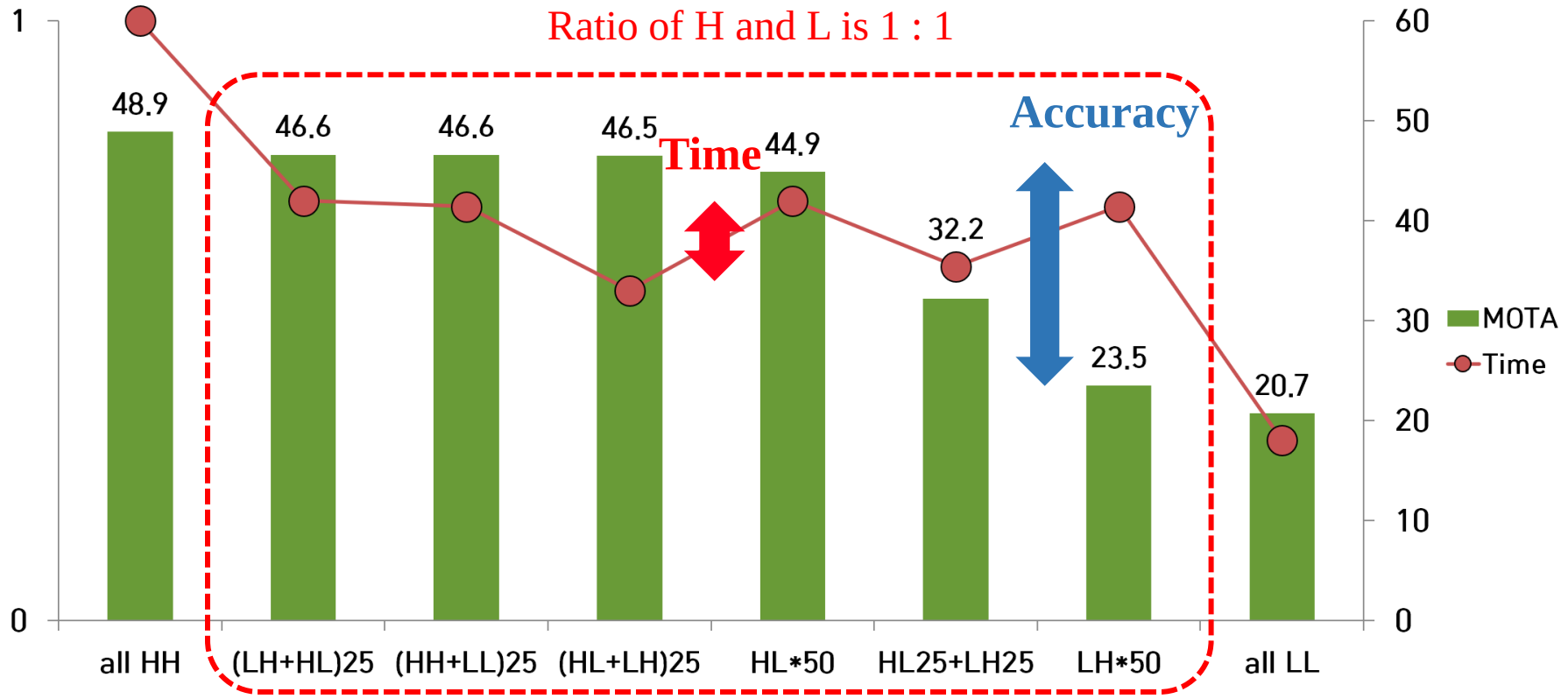
Motivational Observation

- On limited resources
  - Four combinations (HH, HL, LH, LL) are available for trade-off between R1(timing) and R2(accuracy)



Motivational Observation

- Opportunities and challenges of multiple combinations
  - **Opportunities**: the accuracy and time are significantly different according to the detection/tracking combination.



Motivational Observation

- Opportunities and challenges of multiple combinations
  - **Challenges**: How can we find the best combination to satisfy R1 and R2 at runtime?

Jinkyu Lee
Dept. of Computer Science and Engineering

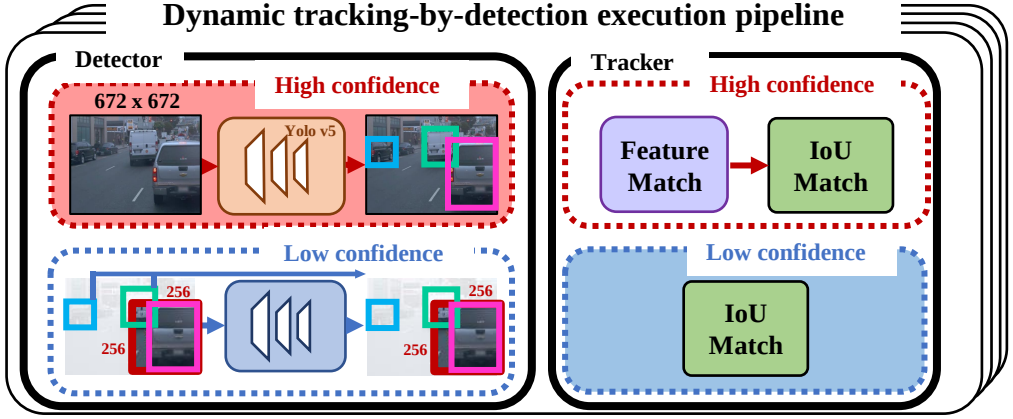
RT-MOT (Real-time Multi-Object Tracking)

- Goal
  - Design a framework for multiple MOT tasks, which achieves (R1) timing guarantee and (R2) accuracy maximization
  - G1: How can we design the system architecture to provide a control knob to explore a trade-off between R1 and R2?
  - G2: How can we efficiently utilize the proposed system architecture to achieve R1 and R2?

RT-MOT (Real-time Multi-Object Tracking)

- G1: How can we design the system architecture to provide a control knob to explore a trade-off between R1 (timing) and R2 (accuracy)?

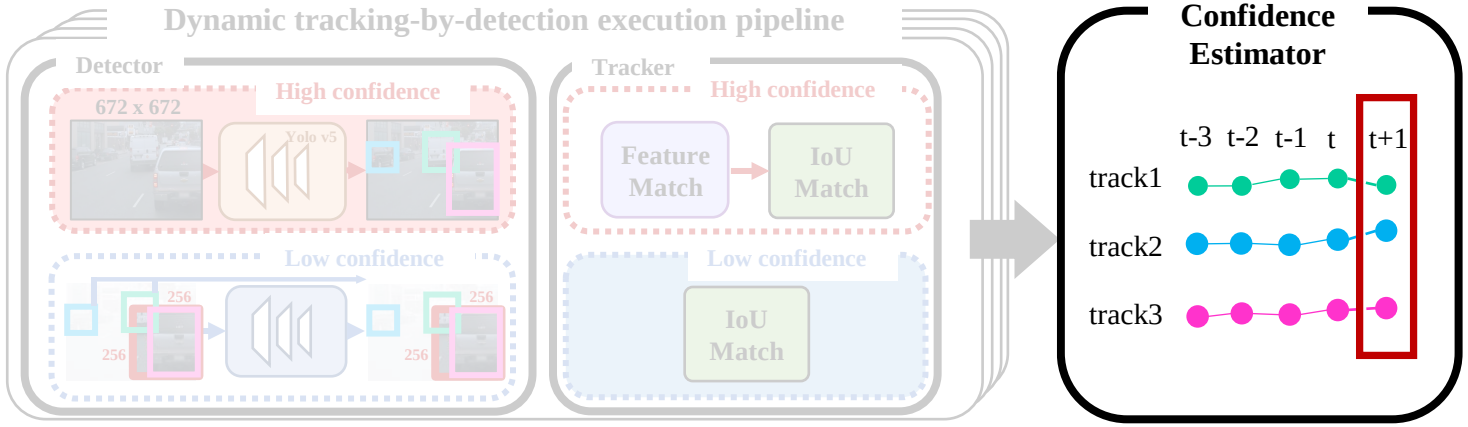
- RT-MOT
  1. **Dynamic tracking-by-detection execution pipeline**: to provide multiple choices of a pair of detection and tracking models



RT-MOT (Real-time Multi-Object Tracking)

- G2: How can we efficiently utilize the proposed system architecture to achieve R1 (timing) and R2 (accuracy)?

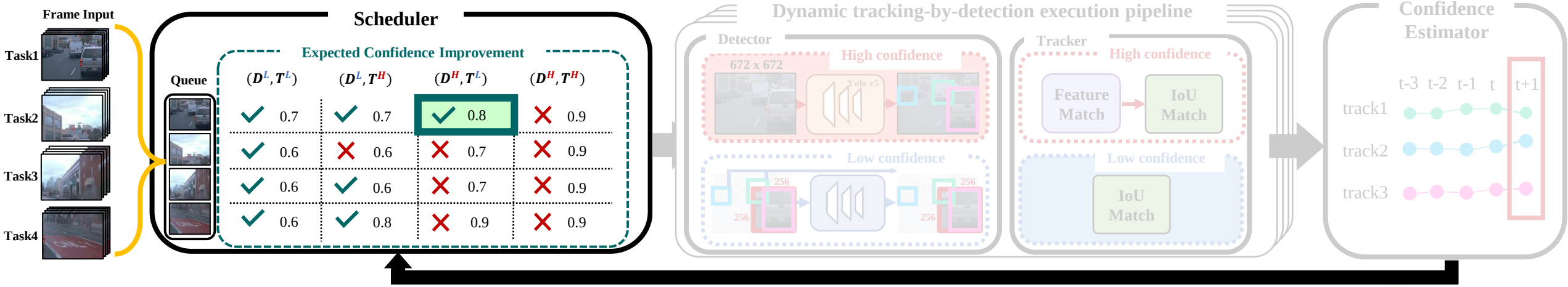
- RT-MOT
  1. **Dynamic tracking-by-detection execution pipeline**: to provide multiple choices of a pair of detection and tracking models
  2. **Confidence estimator**: to capture the relation between the object reliability and R2



RT-MOT (Real-time Multi-Object Tracking)

- G2: How can we efficiently utilize the proposed system architecture to achieve R1 (timing) and R2 (accuracy)?

- RT-MOT
  1. **Dynamic tracking-by-detection execution pipeline**: to provide multiple choices of a pair of detection and tracking models
  2. **Confidence estimator**: to capture the relation between the object reliability and R2
  3. **Frame-level scheduler**: to select the best choice for R2 while satisfying R1



RT-MOT Components

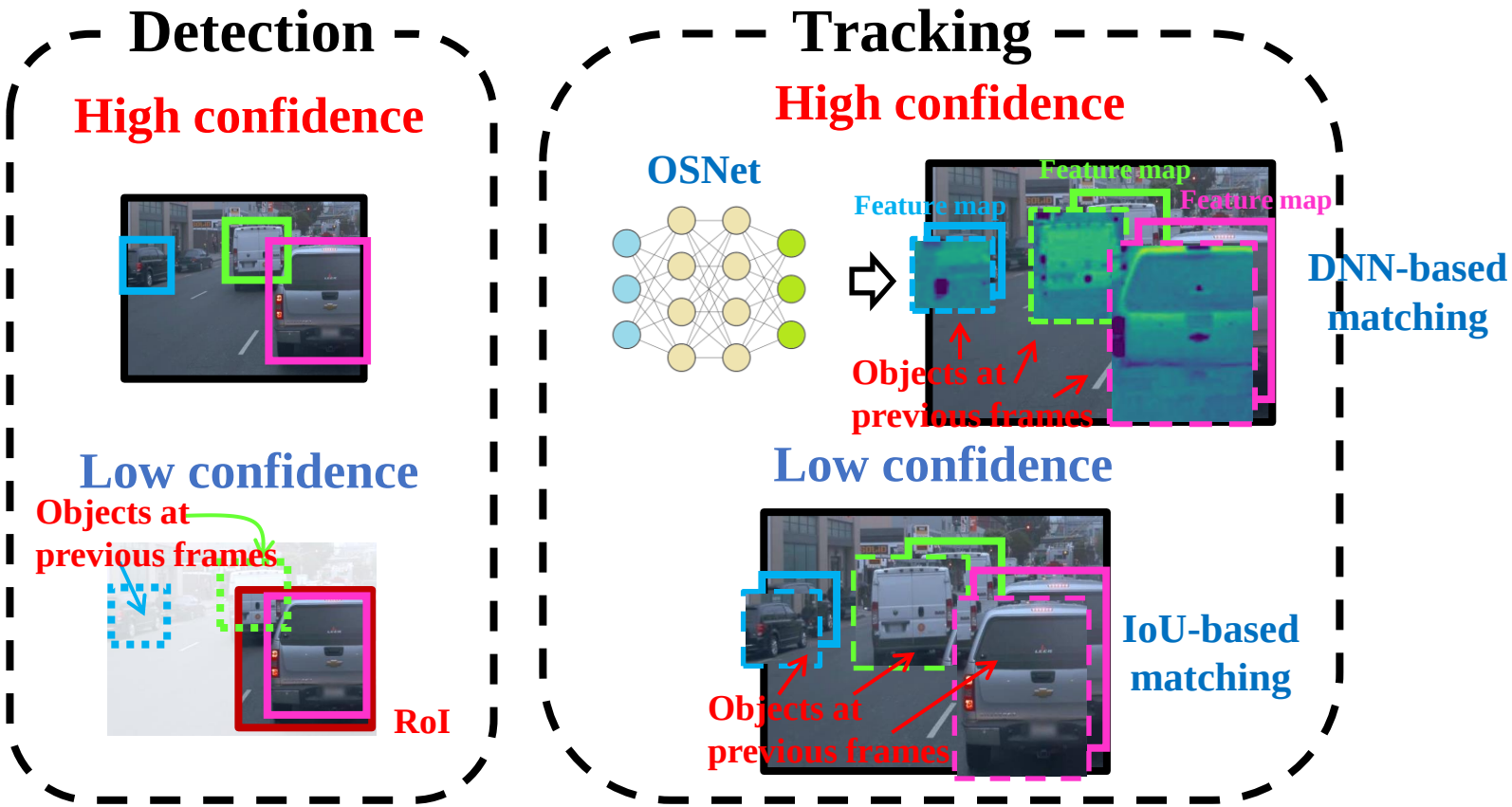
1. Dynamic tracking-by-detection execution pipeline

2. Confidence estimator

3. Frame-level scheduler

RT-MOT Components

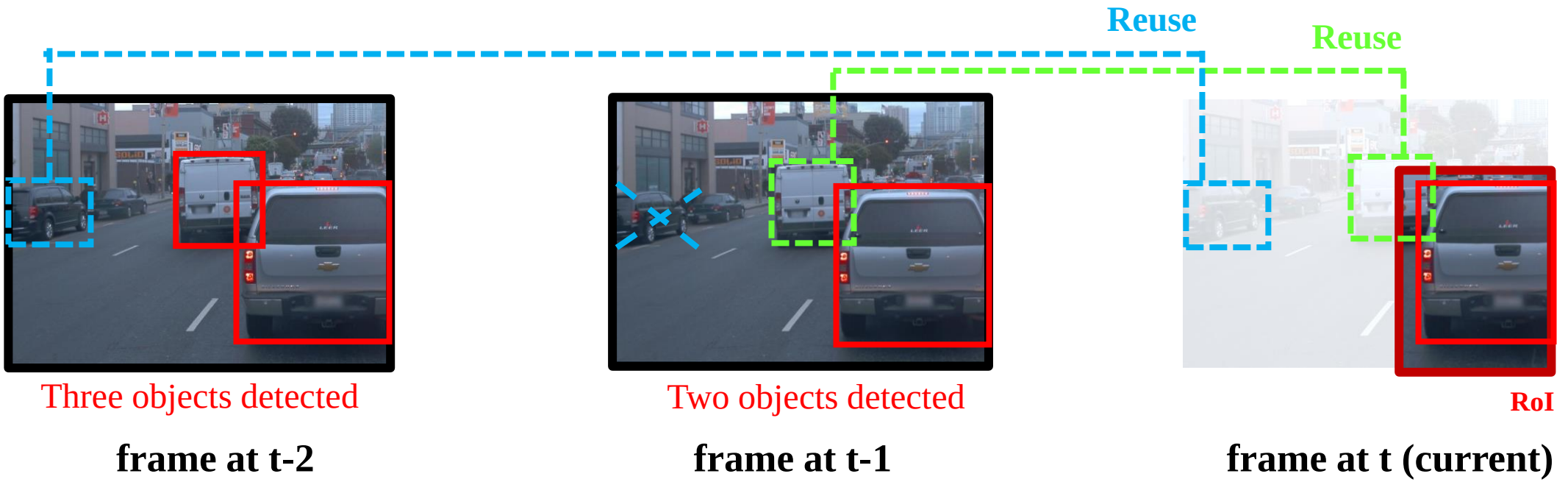
1. Dynamic tracking-by-detection execution pipeline
   - RT-MOT provides multiple choices of a pair of detection and association models



RT-MOT Components

1. Dynamic tracking-by-detection execution pipeline

- Low confidence detection: reuses motion information of objects detected in the previous frames

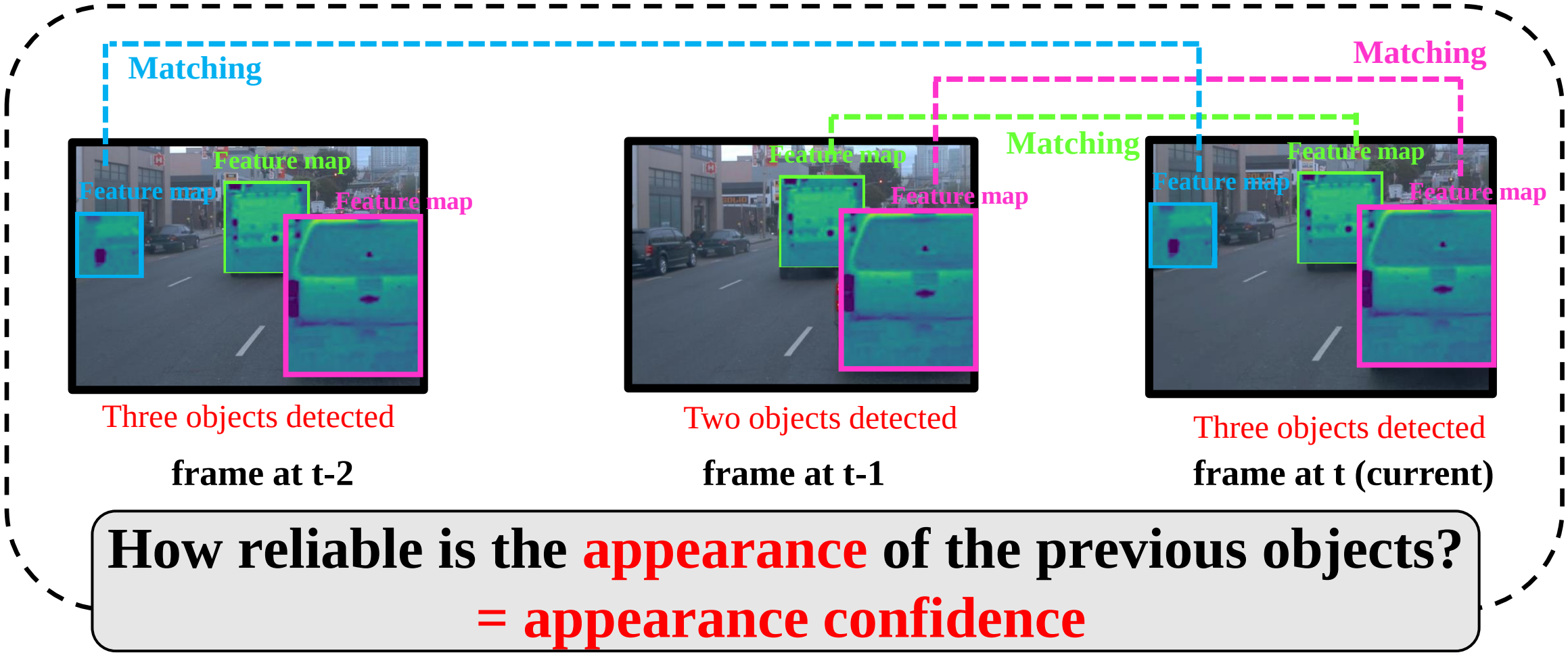


How reliable is the motion of the previous objects?
= motion confidence

RT-MOT Components

1. Dynamic tracking-by-detection execution pipeline

- High confidence tracking: reuses appearance information of objects detected in the previous frames



How reliable is the appearance of the previous objects?
= appearance confidence

RT-MOT Components

2. Multi-object tracking confidence estimator
   - How reliable is the motion (=size, position) of the previous objects? Motion confidence
   - How reliable is the appearance (=feature) of the previous objects? Appearance confidence

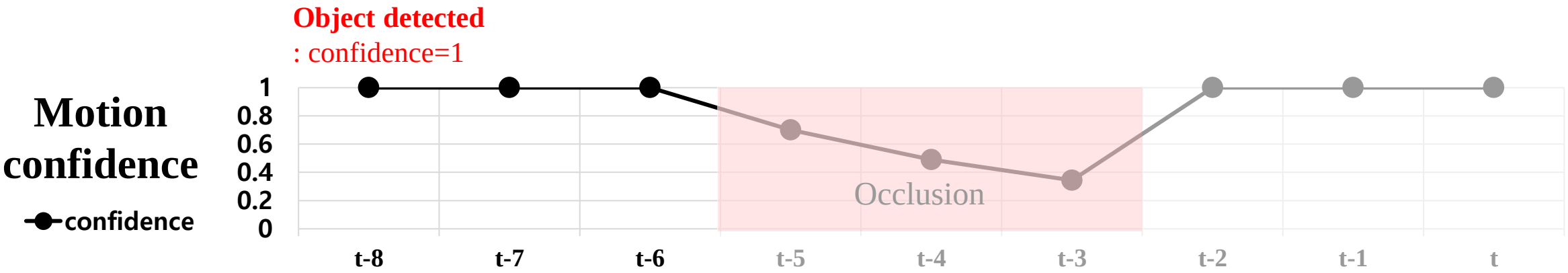
- Q1. How can we update (track) the changing confidence of objects?
  - Confidence update rule

- Q2. How can we estimate confidence improvement according to different selections of detection/tracking models?
  - Confidence estimation rule

Expected accuracy improvement to be used for frame-level scheduler

RT-MOT Components

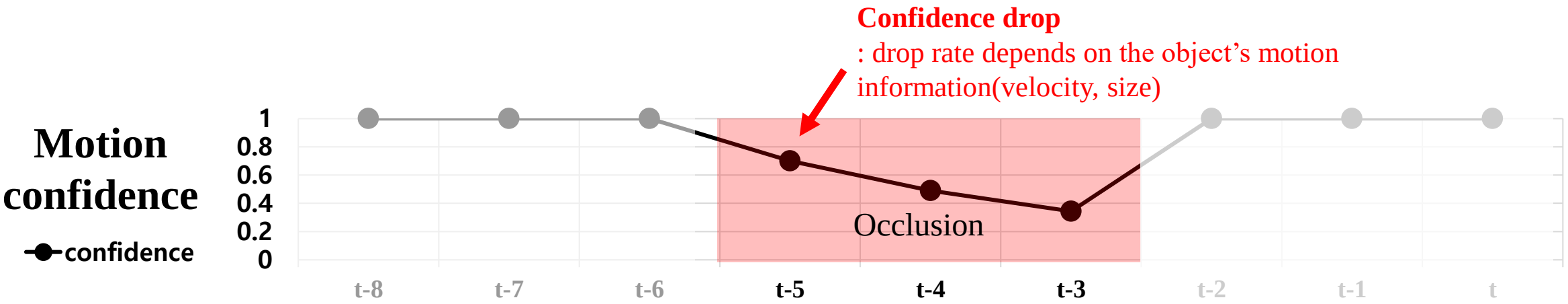
- Q1. How can we update (track) the changing confidence of objects?
  - Confidence update rule

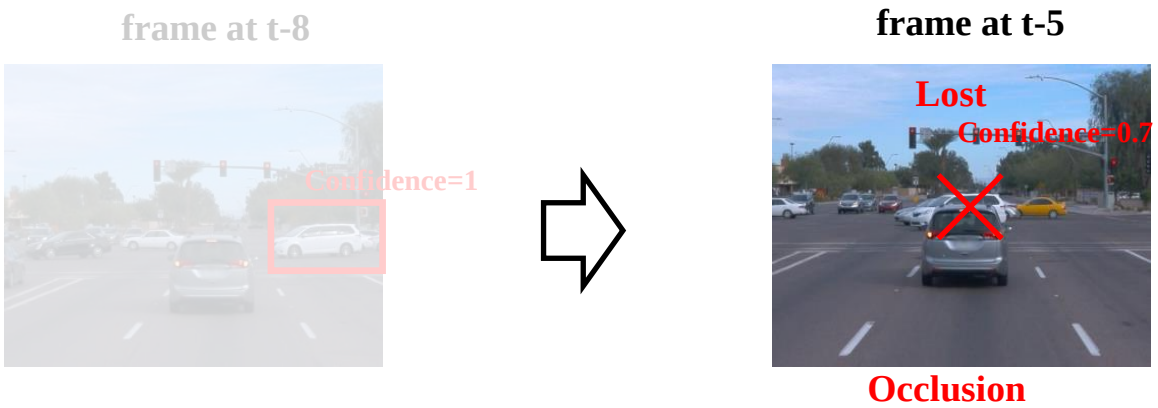


frame at t-8

RT-MOT Components

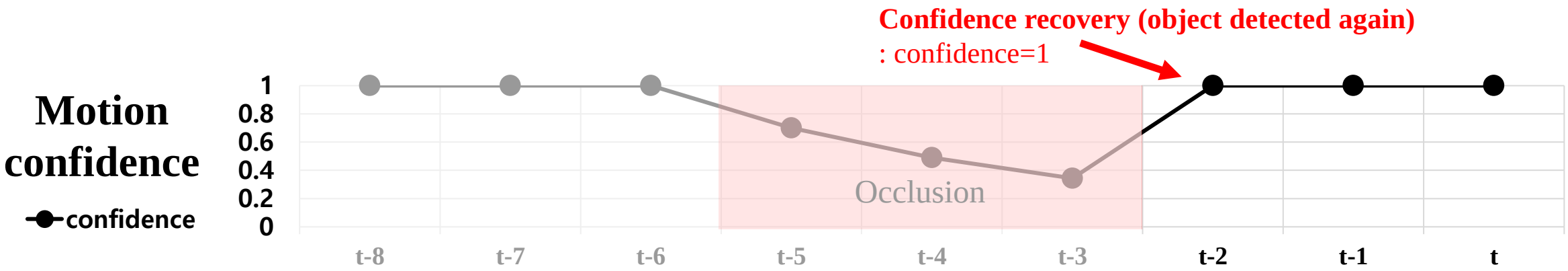
- Q1. How can we update (track) the changing confidence of objects?
  - Confidence update rule





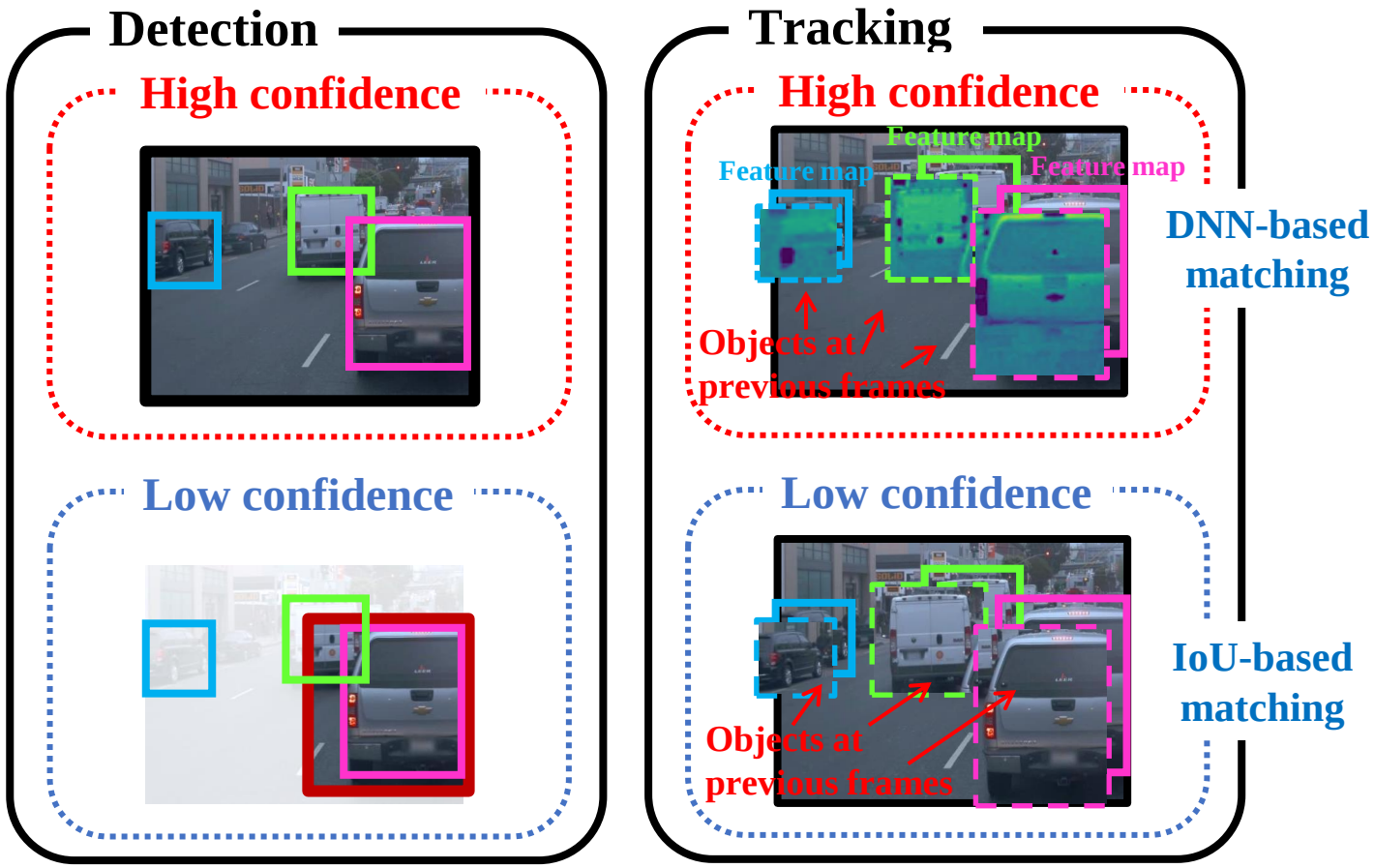
RT-MOT Components

- Q1. How can we update (track) the changing confidence of objects?
  - Confidence update rule



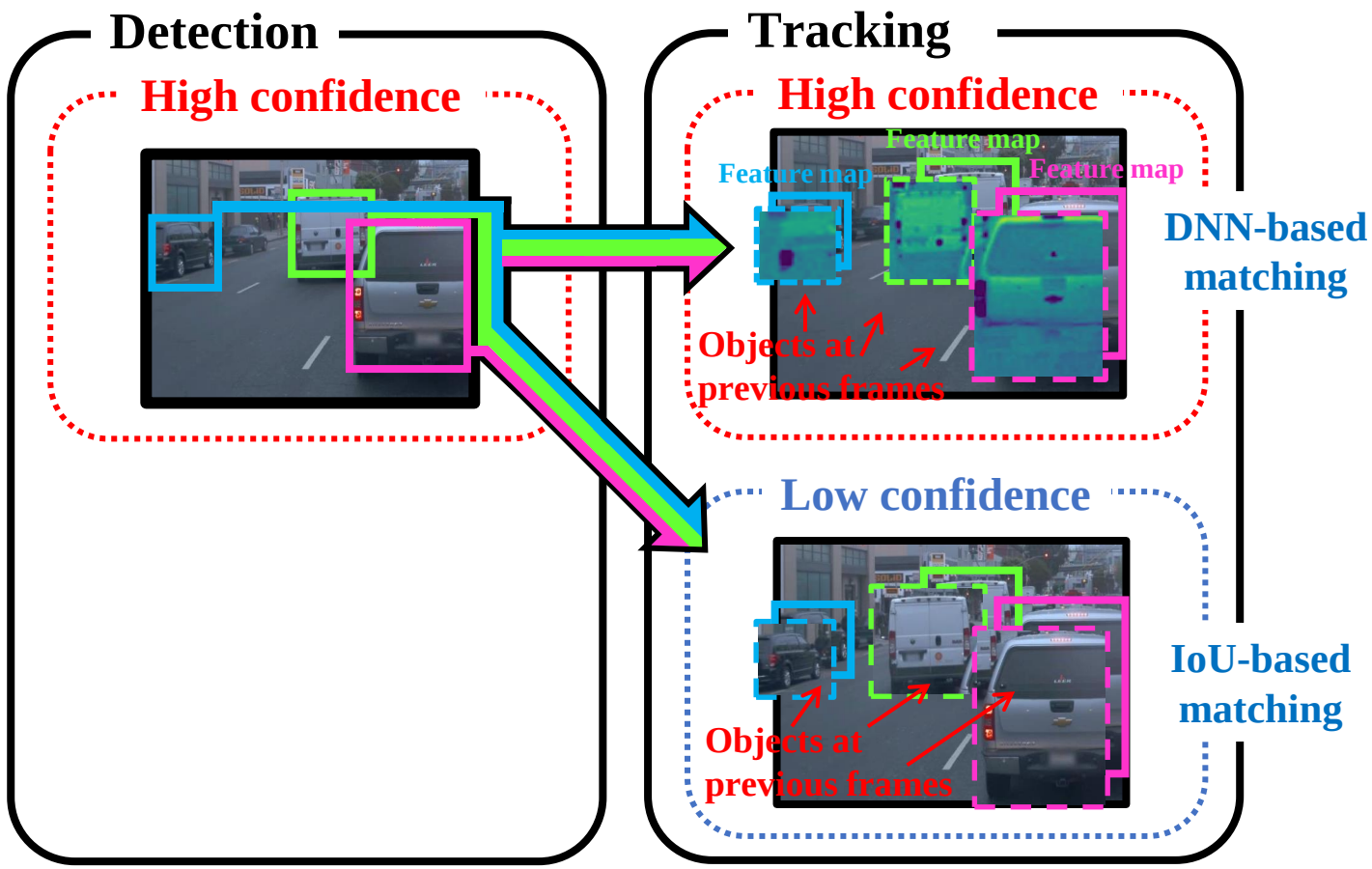
RT-MOT Components

- Q2. How can we estimate confidence improvement according to different selections of detection/tracking models?
  - Confidence estimation rule



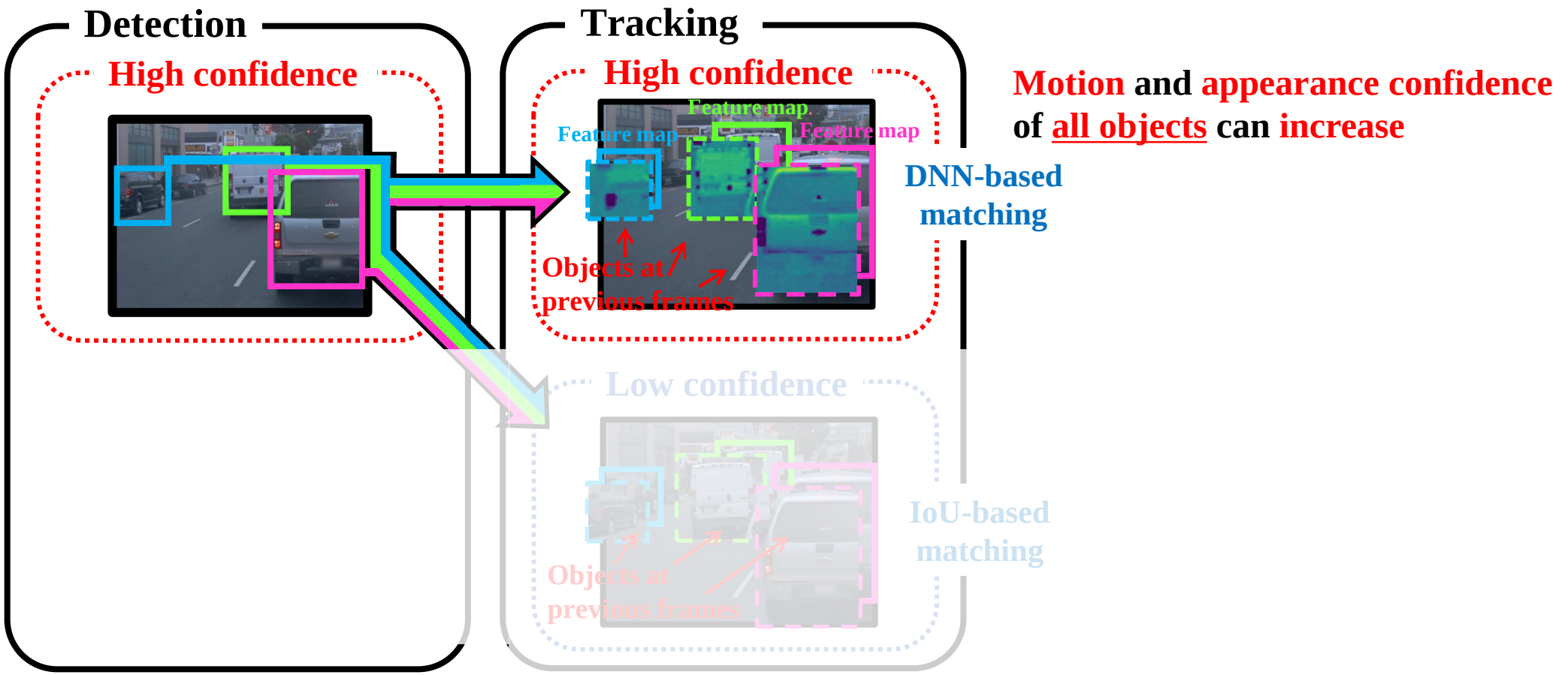
RT-MOT Components

- Q2. How can we estimate confidence improvement according to different selections of detection/tracking models?
  - Confidence estimation rule



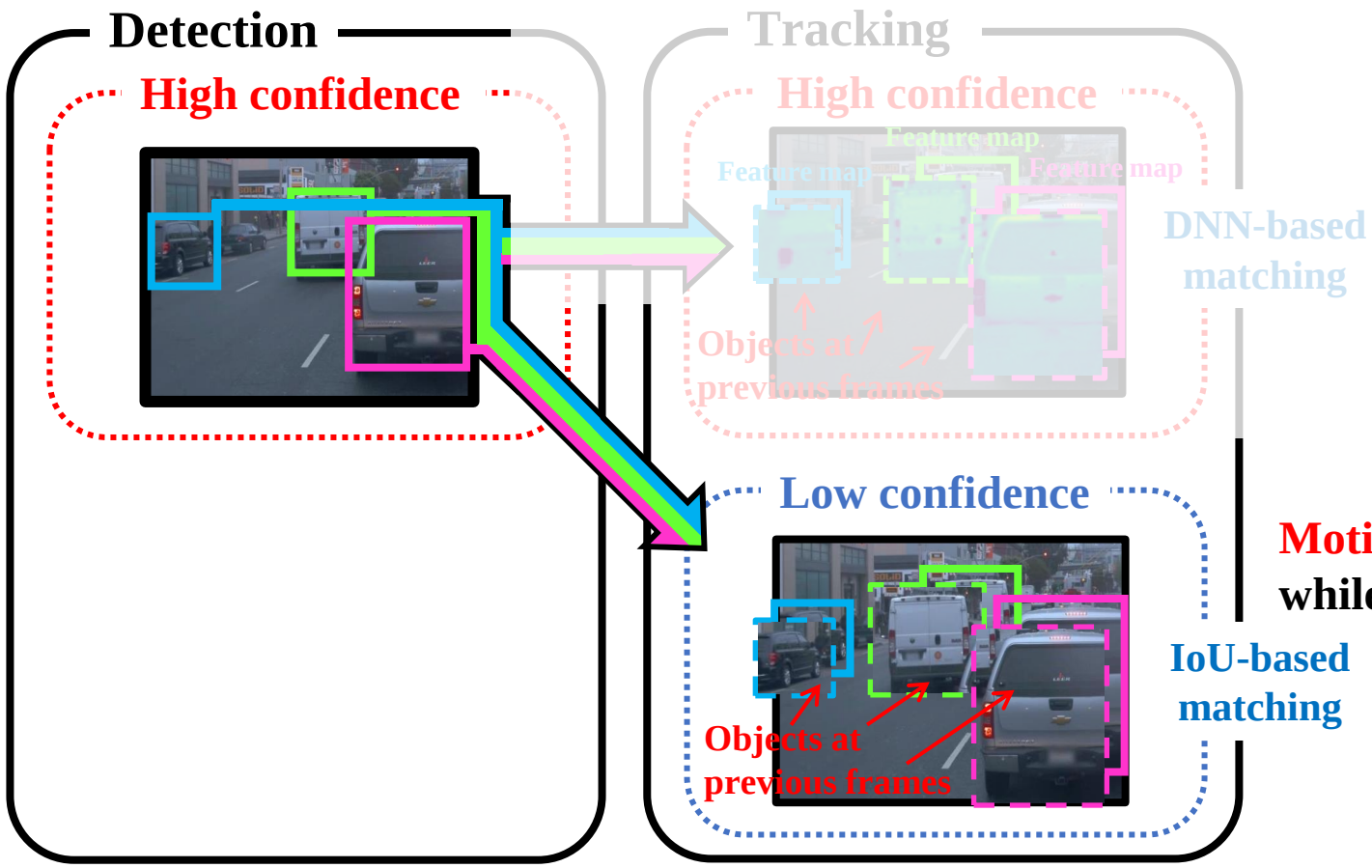
RT-MOT Components

- Q2. How can we estimate confidence improvement according to different selections of detection/tracking models?
  - Confidence estimation rule



RT-MOT Components

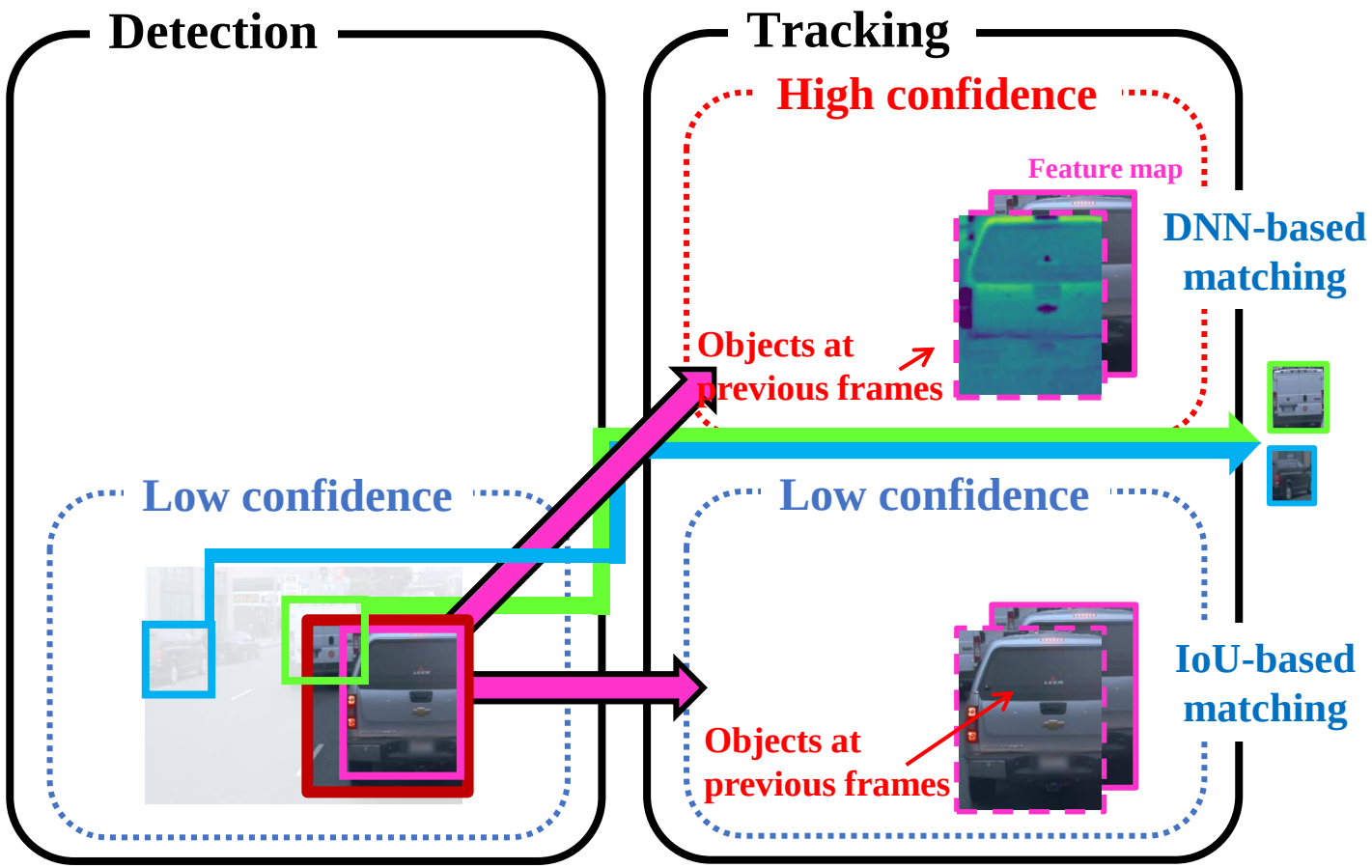
- Q2. How can we estimate confidence improvement according to different selections of detection/tracking models?
  - Confidence estimation rule



Motion confidence of all objects increases while the appearance confidence of them can decrease

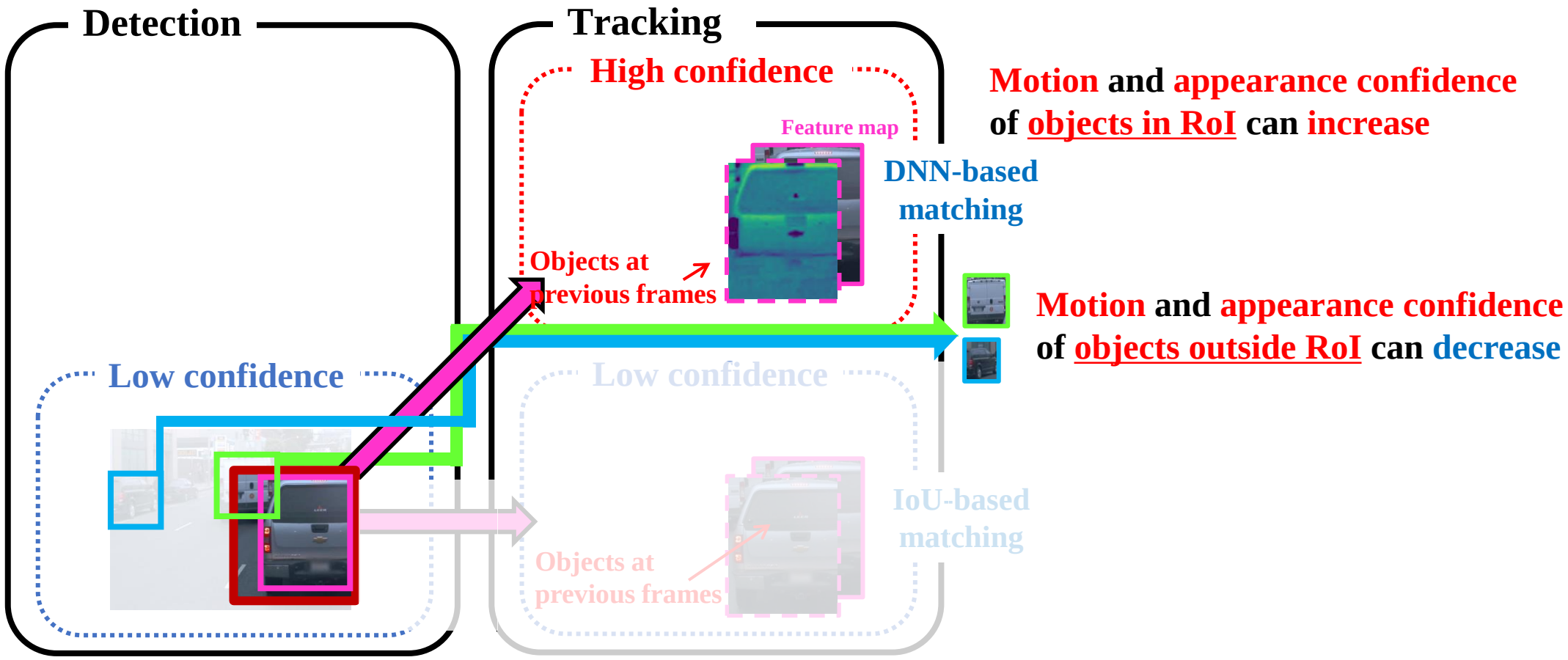
RT-MOT Components

- Q2. How can we estimate confidence improvement according to different selections of detection/tracking models?
  - Confidence estimation rule



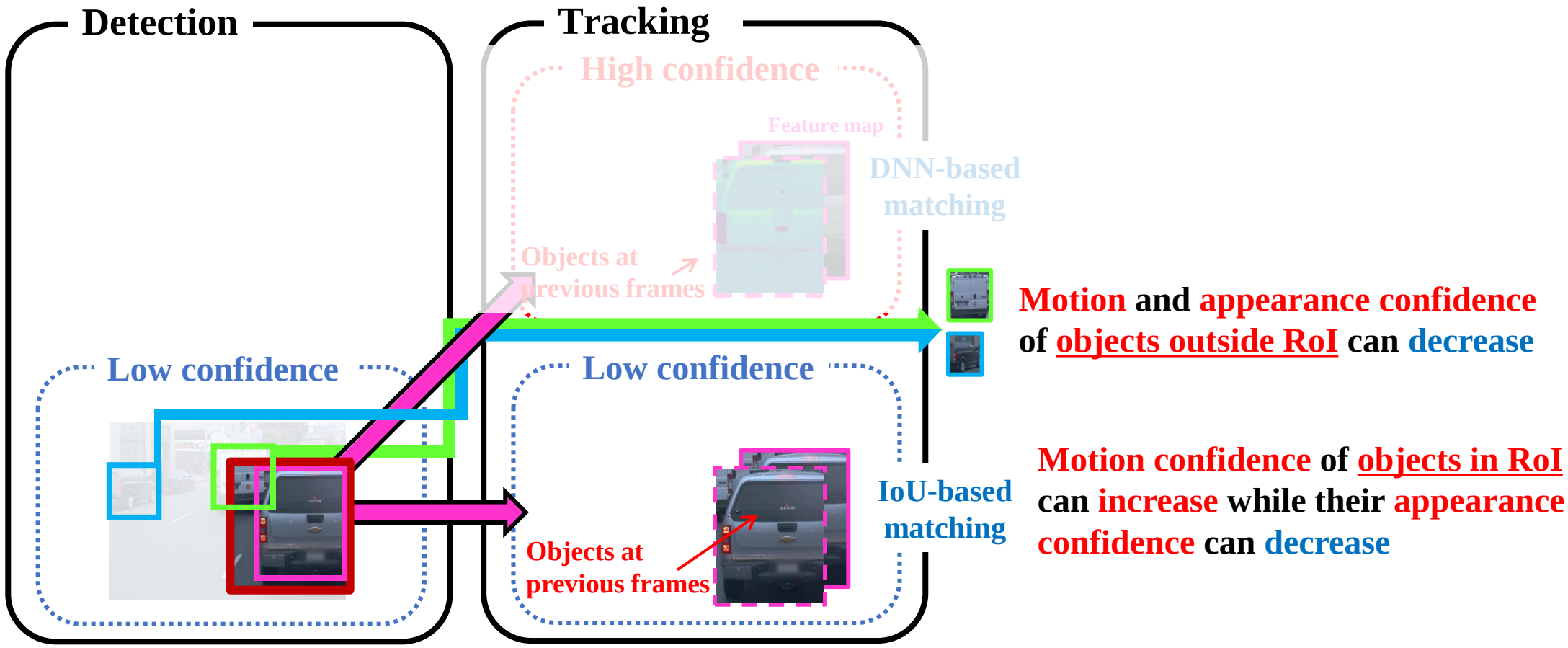
RT-MOT Components

- Q2. How can we estimate confidence improvement according to different selections of detection/tracking models?
  - Confidence estimation rule



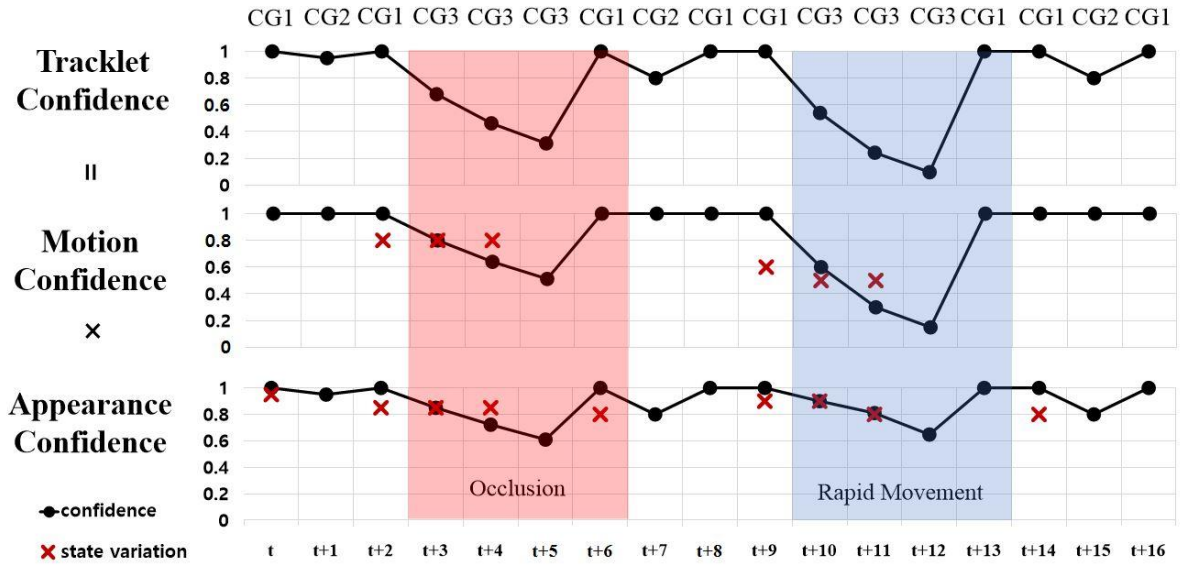
RT-MOT Components

- Q2. How can we estimate confidence improvement according to different selections of detection/tracking models?
  - Confidence estimation rule



RT-MOT Components

- Q2. How can we estimate confidence improvement according to different selections of detection/tracking models?



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

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

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

where

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

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

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

RT-MOT Components

3. Frame-level flexible scheduler, $NPFP^{flex}$
   - Non-preemptive fixed-priority scheduling for multiple MOT tasks
   - Confidence-aware detection/tracking model selection

$NPFP^{flex}$ with objectives

O1. Providing offline timing guarantee

O2. Maximizing tracking accuracy at run time while not compromising timing guarantee

NPFP$^{\text{flex}}$

- O1. Providing offline timing guarantee
- O2. Maximizing tracking accuracy at run time while not compromising timing guarantee

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

NPFP^{flex}

- O1. Providing offline timing guarantee

Theorem 1:
If the following equation holds, τ is schedulable by NPFP^{flex} with minimum execution $(\boldsymbol{D}^{\boldsymbol{L}}, \boldsymbol{T}^{\boldsymbol{L}})$.

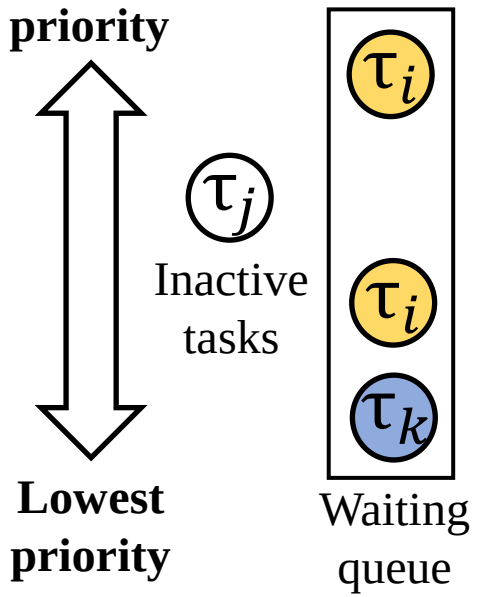
$$R_i \leq T_i \qquad R_i(x+1) = C_i^{LL} + \max_{\tau_j \in LP(\tau_i)} C_j^{LL} + \sum_{\tau_h \in HP(\tau_i)} \left\lceil \frac{R_i(x)}{T_h} \right\rceil \cdot C_h^{LL}$$

$C_i^{(L\ or\ H)\ (L\ or\ H)}$: WCET of τ_i with (L or H)-confidence detection and (L or H)-confidence association
T_i : Period of τ_i
R_i : the worst-case response time of τ_i

NPFP^{flex}

- O2. Maximizing tracking accuracy at run time while not compromising timing guarantee
  - At every scheduling decision, NPFP^{flex}

1. finds schedulable pairs of detection and tracking models for every active task

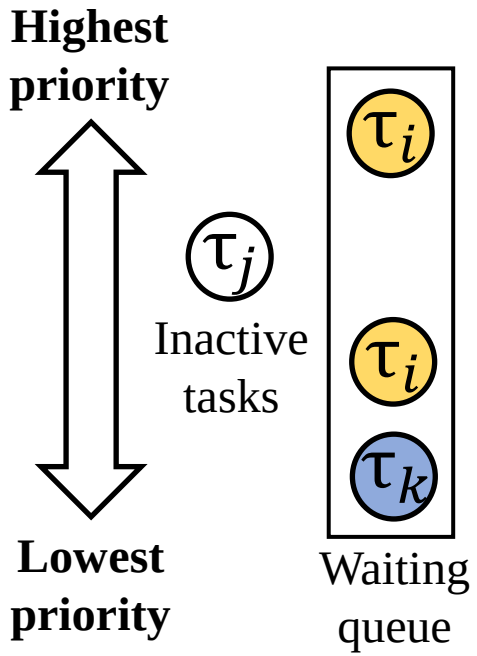


Expected Confidence Improvement

(D^L, T^L)	(D^L, T^H)	(D^H, T^L)	(D^H, T^H)
✓	✓	✓	✗
✓	✓	✗	✗
✓	✓	✓	✗

NPFP^{flex}

- O2. Maximizing tracking accuracy at run time while not compromising timing guarantee
  - At every scheduling decision, NPFP^{flex}
  1. finds schedulable pairs of detection and tracking models for every active task
  2. then, selects the best confidence improvement among schedulable pairs

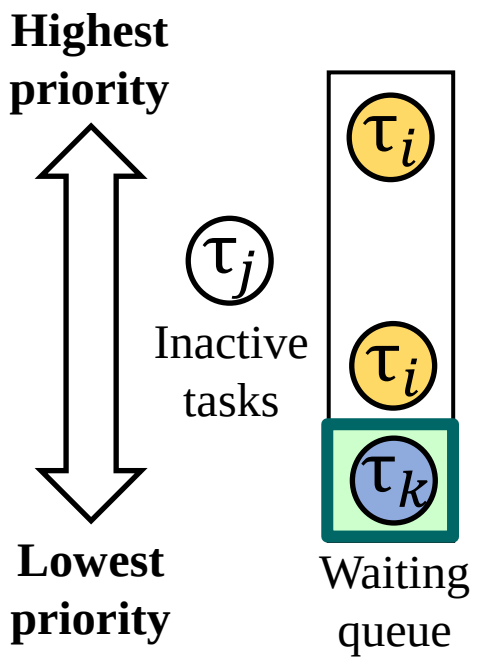


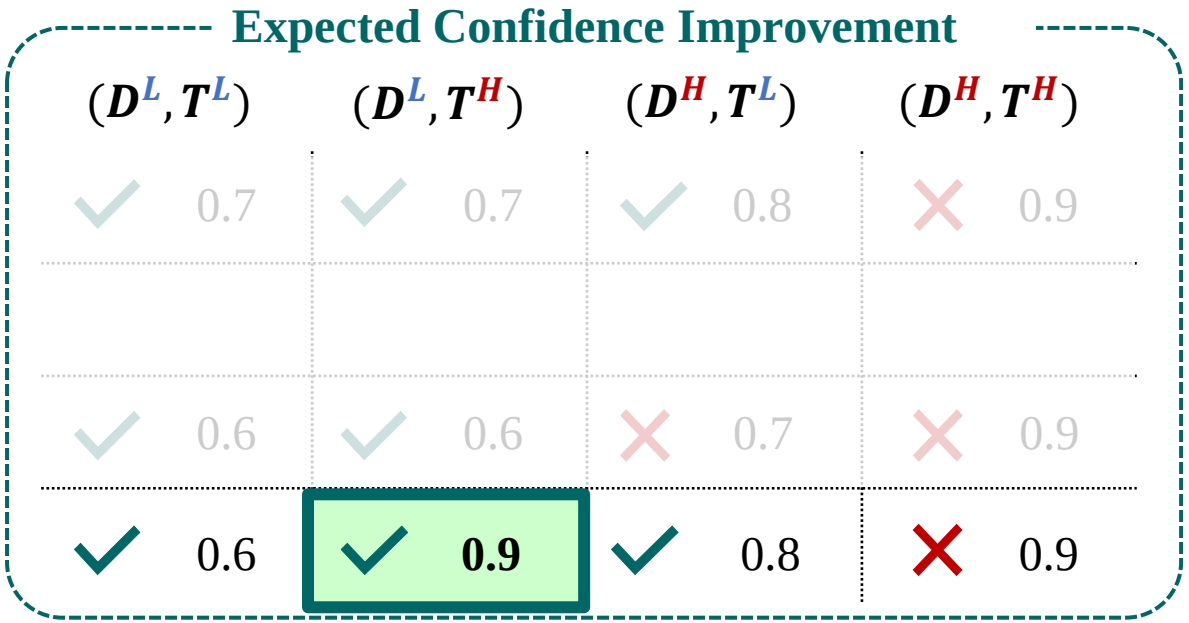
Expected Confidence Improvement

(D^L, T^L)	(D^L, T^H)	(D^H, T^L)	(D^H, T^H)
✓ 0.7	✓ 0.7	✓ 0.8	✗ 0.9
✓ 0.6	✓ 0.6	✗ 0.7	✗ 0.9
✓ 0.6	✓ 0.9	✓ 0.8	✗ 0.9

NPFP^{flex}

- O2. Maximizing tracking accuracy at run time while not compromising timing guarantee
  - At every scheduling decision, NPFP^{flex}
  1. finds schedulable pairs of detection and tracking models for every active task
  2. then, selects the best confidence improvement among schedulable pairs

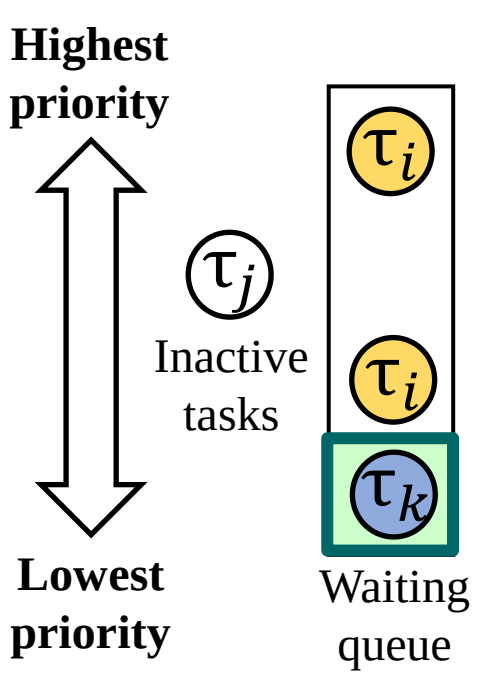




NPFP^{flex}

- O2. Maximizing tracking accuracy at run time while not compromising timing guarantee
  - At every scheduling decision, NPFP^{flex}
    1. finds schedulable pairs of detection and tracking models for every active task
    2. then, selects the best confidence improvement among schedulable pairs

How to guarantee schedulability of every task at runtime?

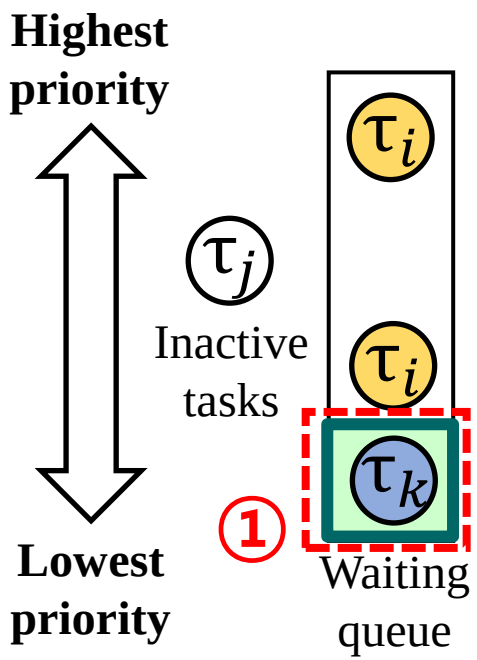


Expected Confidence Improvement

(D^L, T^L)	(D^L, T^H)	(D^H, T^L)	(D^H, T^H)
✓ 0.7	✓ 0.7	✓ 0.8	✗ 0.9
✓ 0.6	✓ 0.6	✗ 0.7	✗ 0.9
✓ 0.6	✓ **0.9**	✓ 0.8	✗ 0.9

NPFP^{flex}

- Online timing guarantee for
  1. Selected active task τ_k

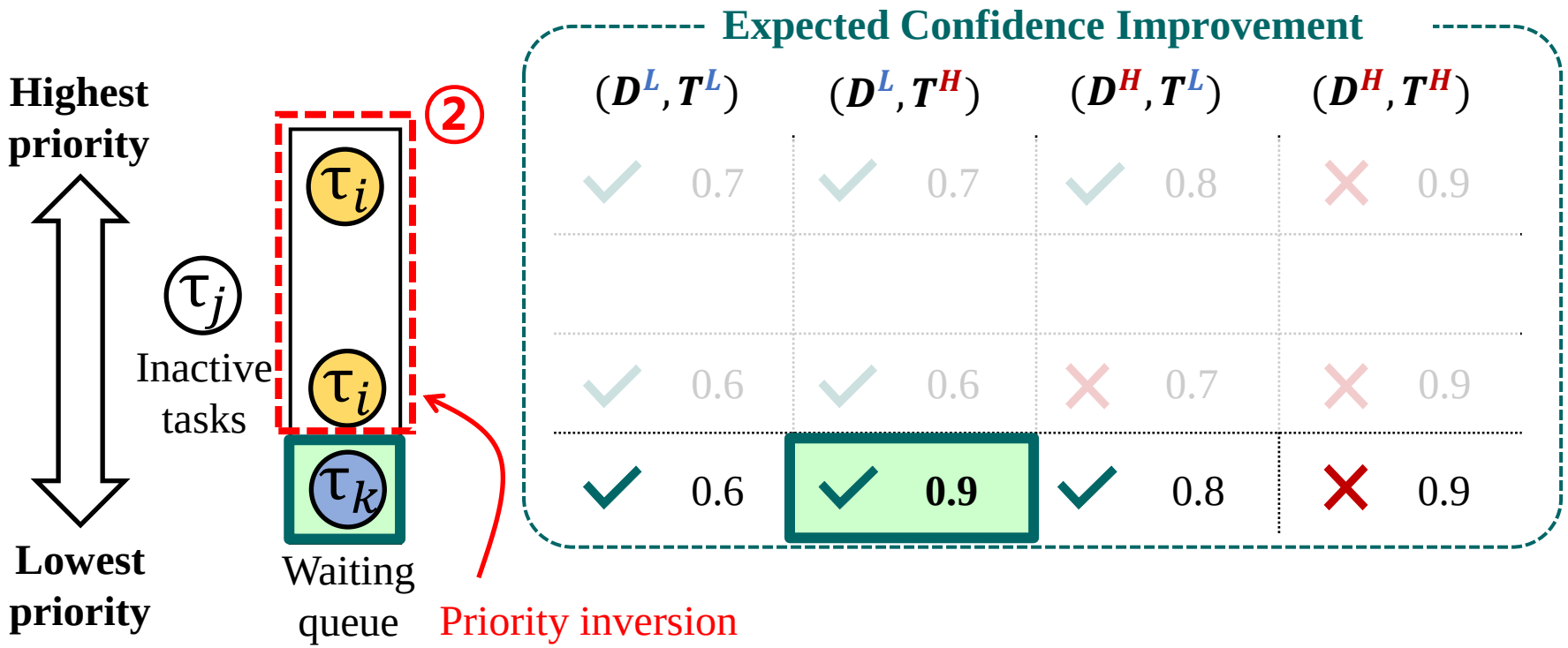


Expected Confidence Improvement

(D^L, T^L)	(D^L, T^H)	(D^H, T^L)	(D^H, T^H)
✓ 0.7	✓ 0.7	✓ 0.8	✗ 0.9
✓ 0.6	✓ 0.6	✗ 0.7	✗ 0.9
✓ 0.6	✓ **0.9**	✓ 0.8	✗ 0.9

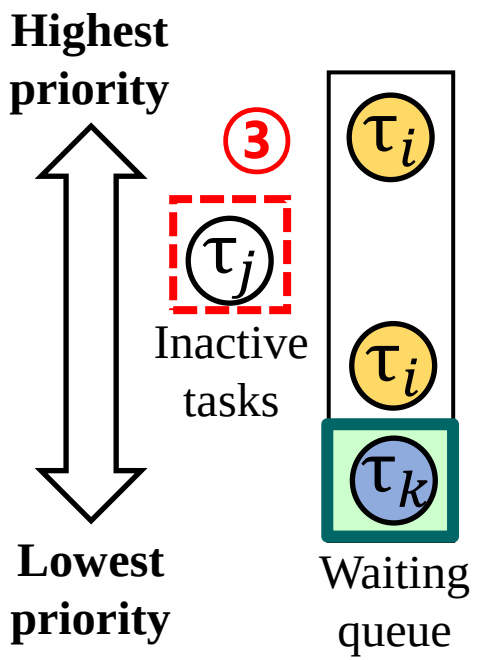
NPFP^{flex}

- Online timing guarantee for
  1. Selected active task τ_k
  2. Not selected active task τ_i



NPFP^{flex}

- Online timing guarantee for
  1. Selected active task τ_k
  2. Not selected active task τ_i
  3. Inactive task τ_j

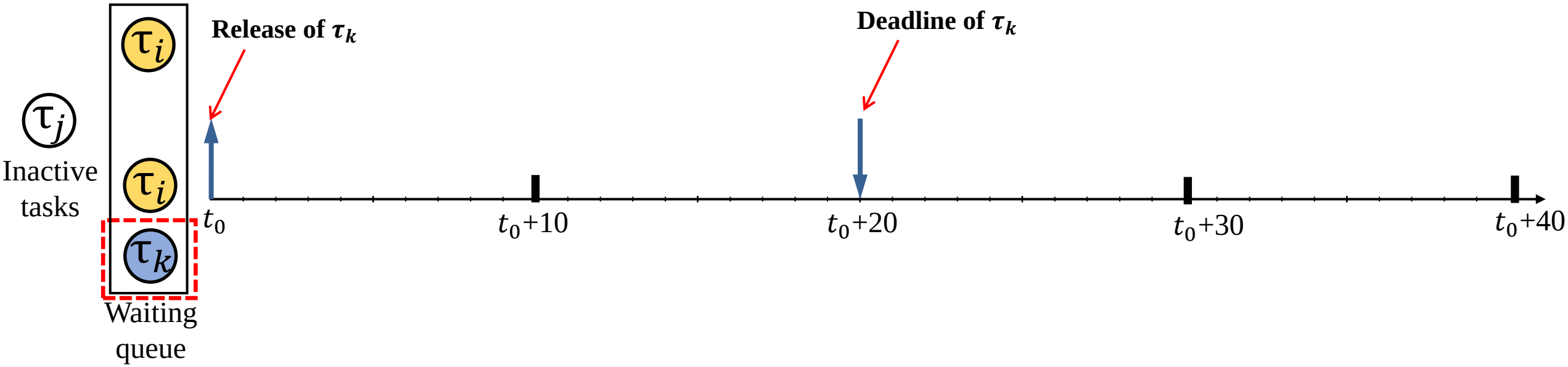


Expected Confidence Improvement

(D^L, T^L)	(D^L, T^H)	(D^H, T^L)	(D^H, T^H)
✓ 0.7	✓ 0.7	✓ 0.8	✗ 0.9
✓ 0.6	✓ 0.6	✗ 0.7	✗ 0.9
✓ 0.6	✓ **0.9**	✓ 0.8	✗ 0.9

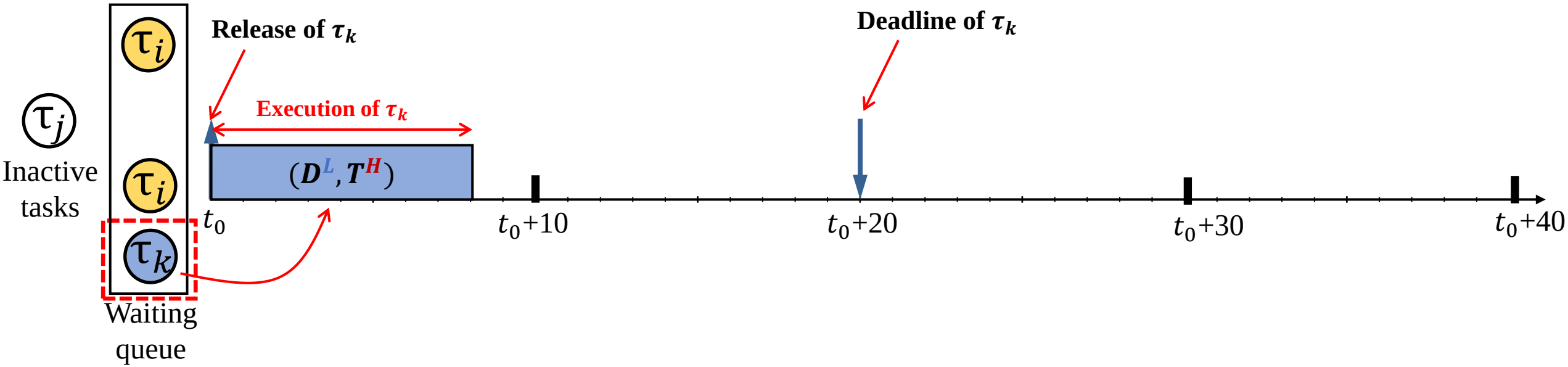
Online Timing Guarantee on NPFP^{flex}

1. Task τ_k selected at t_0



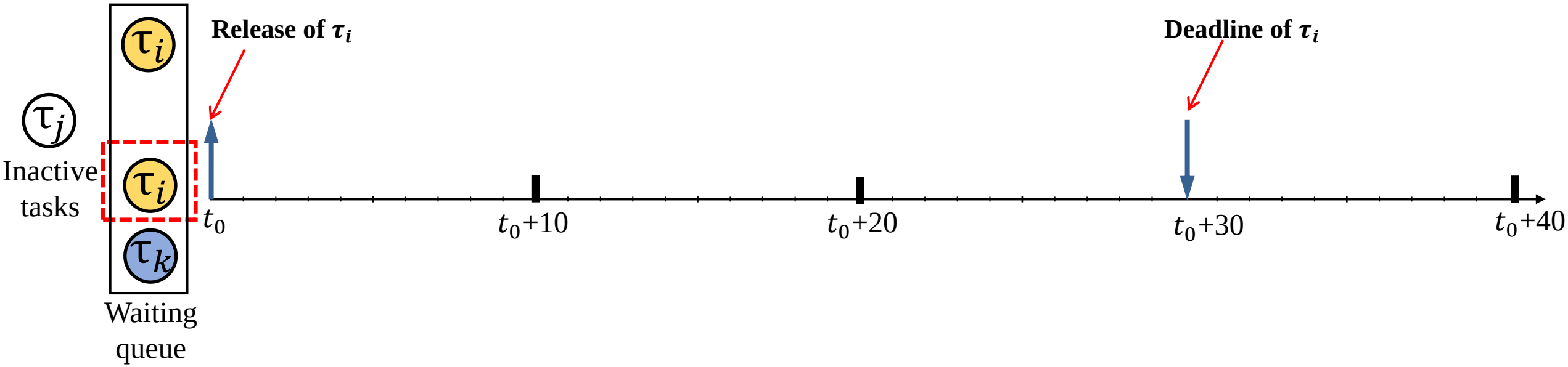
Online Timing Guarantee on NPFP^{flex}

1. Task τ_k selected at t_0



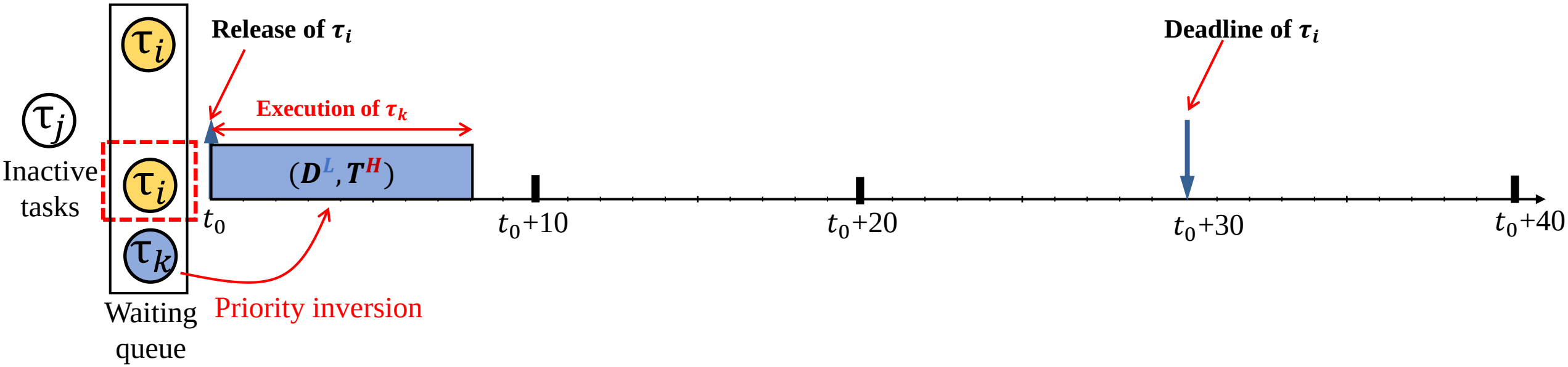
Online Timing Guarantee on NPFP^{flex}

2. Active task(s) τ_i not selected at t_0



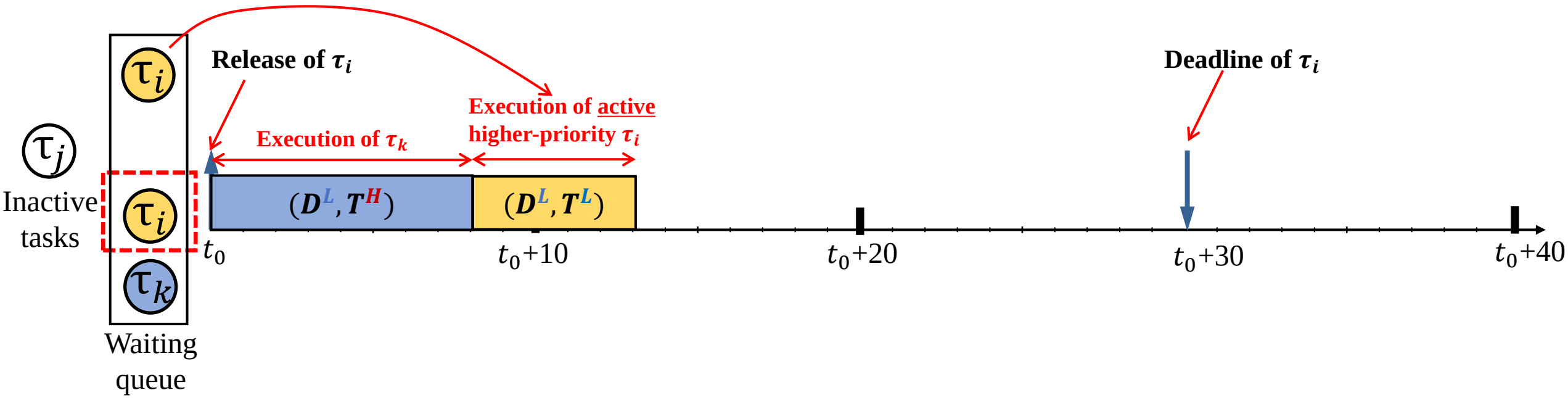
Online Timing Guarantee on NPFP^{flex}

2. Active task(s) τ_i not selected at t_0



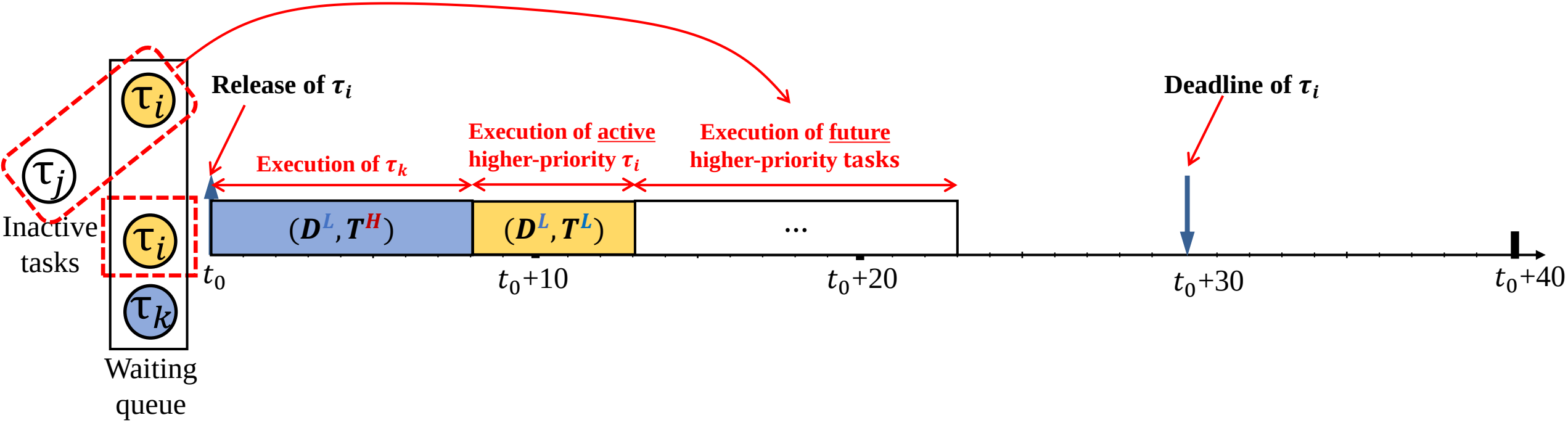
Online Timing Guarantee on NPFP^{flex}

2. Active task(s) τ_i not selected at t_0



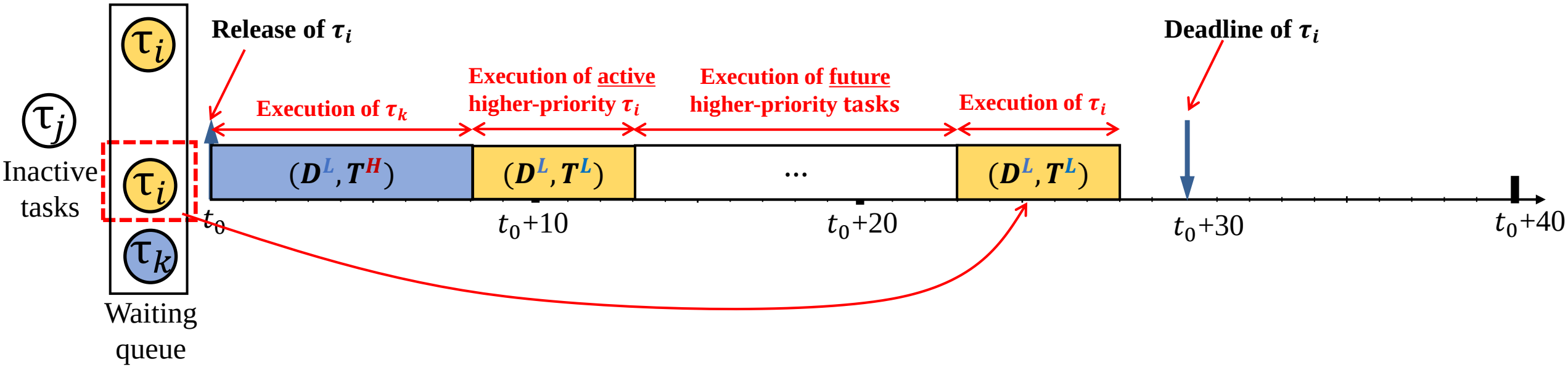
Online Timing Guarantee on NPFP^{flex}

2. Active task(s) τ_i not selected at t_0



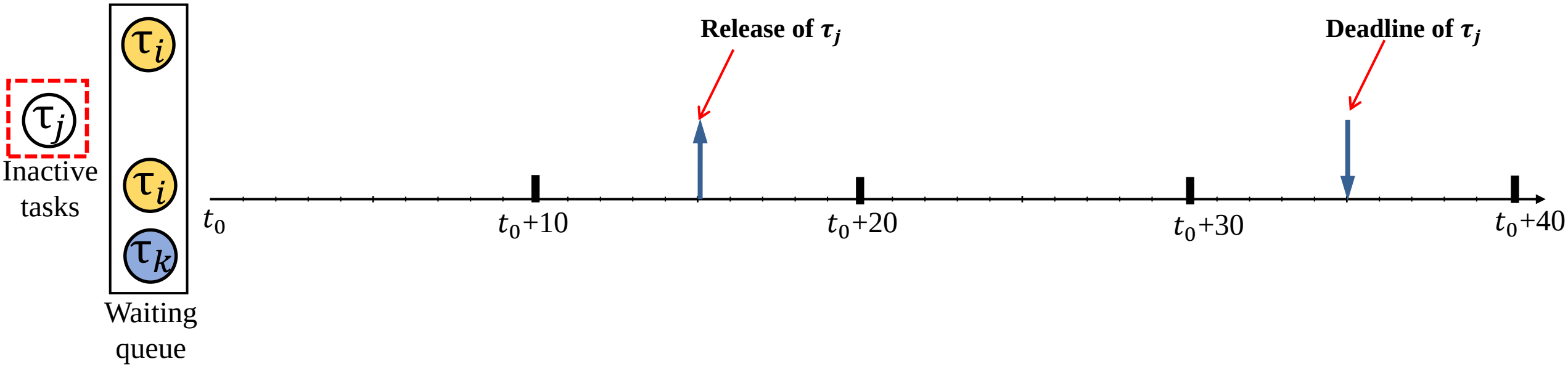
Online Timing Guarantee on NPFP^{flex}

2. Active task(s) τ_i not selected at t_0



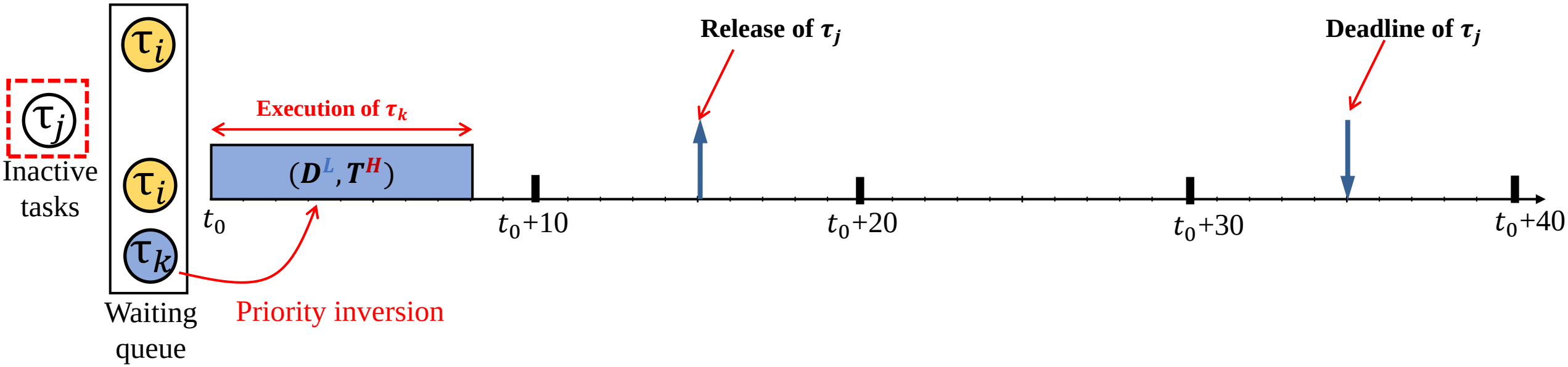
Online Timing Guarantee on NPFP^{flex}

3. Inactive task(s) τ_j



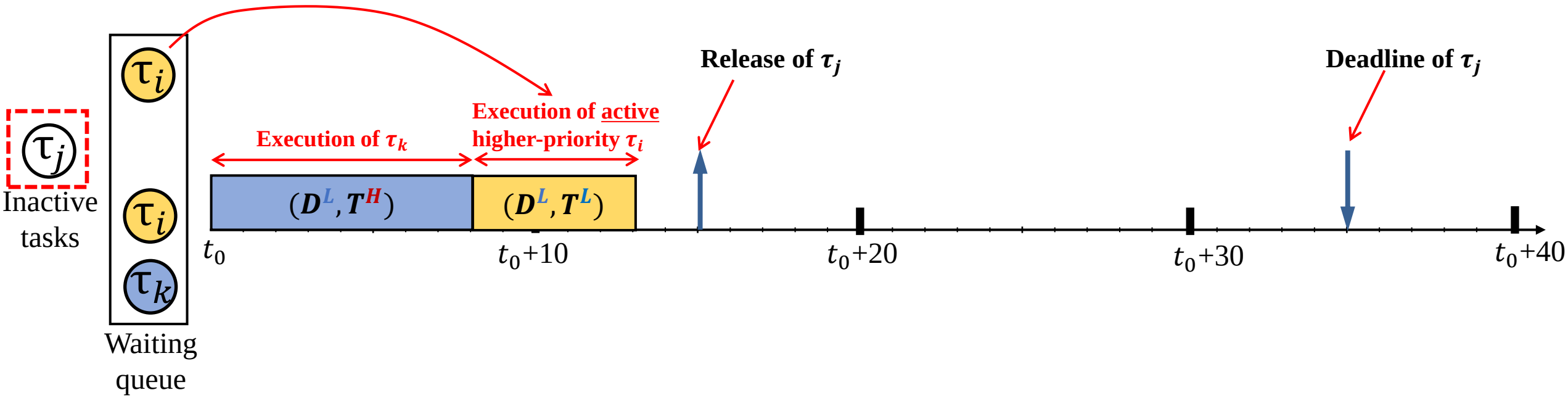
Online Timing Guarantee on NPFP^{flex}

3. Inactive task(s) τ_j



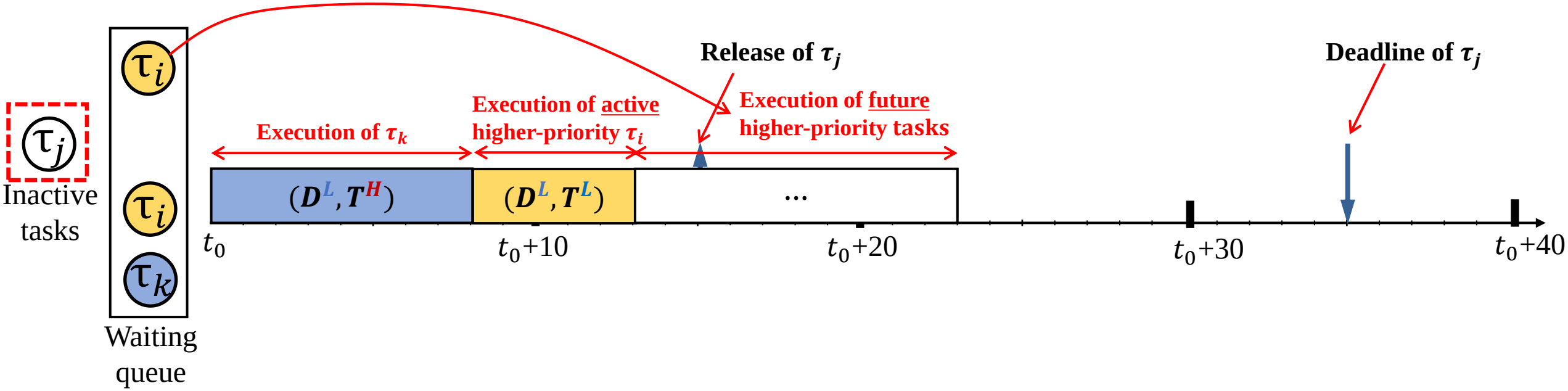
Online Timing Guarantee on NPFP^{flex}

3. Inactive task(s) τ_j



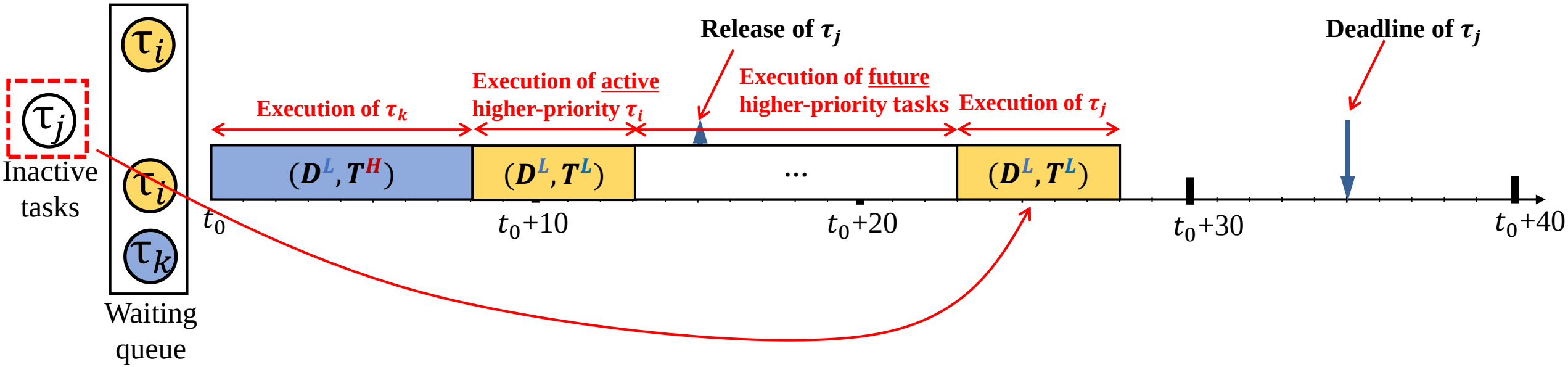
Online Timing Guarantee on NPFP^{flex}

3. Inactive task(s) τ_j



Online Timing Guarantee on NPFP^{flex}

3. Inactive task(s) τ_j



NPFP$^{\text{flex}}$

- Overall run-time mechanism

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

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

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

1. Selected active task τ_k

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

2. Not selected active task τ_i

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

3. Inactive task τ_j

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

NPFP^{flex}

- Overall offline timing guarantees

Theorem 1:
If the following equation holds, τ is schedulable by NPFP^{flex} with minimum execution $(\boldsymbol{D}^{\boldsymbol{L}}, \boldsymbol{T}^{\boldsymbol{L}})$.

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

$C_i^{(L\ or\ H)\ (L\ or\ H)}$: WCET of τ_i with (L or H)-confidence detection and (L or H)-confidence association
T_i : Period of τ_i
R_i : the worst-case response time of τ_i

NPFP^{flex}

Overall mechanism

- Design run-time mechanism such that
  - It does not compromise offline schedulability analysis, and
  - It maximizes the objective (overall accuracy)

Algorithm 1 The NPFP^{flex} scheduling algorithm

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

```
flex ← F
for Every active job (denoted by J_k of τ_k) do
    for C_k ∈ {C_k^LL, C_k^LH, C_k^HL, C_k^HH}, respectively for {(D^L,A^L),(D^L,A^H), (D^H,A^L), (D^H,A^H)} do
        if The following three conditions hold for assigned C_k: (i) Eq. (11) holds; (ii) Eq. (12) holds for all τ_j ∈ τ with Z_j(t_0) = T, and (iii) Eq. (13) holds for all τ_j ∈ τ with Z_j(t_0) = F then
            Calculate ΔΩ̄_{τ_k} in Eq. (8) for given (D^{L/H},A^{L/H})
            flex ← T
        end if
    end for
end for
if flex=T then
    Execute J_k for C_k, whose ΔΩ̄_{τ_k} is the largest.
else
    Execute the job of τ_i, whose priority is the highest among all the jobs active at t_0, during at most C_i^LL for (D^L,A^L), which is the same as NPFP^min.
end if
```

1. Selected active task τ_k

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

2. Not selected active task τ_i

$$C_j^{LL} + C_k + \sum_{\tau_h \in HP(\tau_j)\setminus\{\tau_k\} | Z_h(t_0)=T} C_h^{LL} + \sum_{\tau_h \in HP(\tau_j) | r_h(t_0) < r_j(t_0)} \left\lceil \frac{r_j(t_0) - r_h(t_0)}{T_h} \right\rceil \cdot C_h^{LL} \le r_j(t_0) - t_0 \tag{12}$$

3. Inactive task τ_j

$$C_j^{LL} + C_k + \sum_{\tau_h \in HP(\tau_j)\setminus\{\tau_k\} | Z_h(t_0)=T} C_h^{LL} + \sum_{\tau_h \in HP(\tau_j) | r_h(t_0) < r_j(t_0)+T_j} \left\lceil \frac{r_j(t_0) + T_j - r_h(t_0)}{T_h} \right\rceil \cdot C_h^{LL} \le r_j(t_0) + T_j - t_0 \tag{13}$$

Theorem 1:
If the following equation holds, τ is schedulable by NPFP^{flex} with minimum execution $(\boldsymbol{D}^L, \boldsymbol{T}^L)$.

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

Jinkyu Lee
Dept. of Computer Science and Engineering

NPFP$^{\text{flex}}$

- O1. Providing offline timing guarantee
- O2. Maximizing tracking accuracy at run time while not compromising timing guarantee

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

RT-MOT Evaluation

- Evaluation settings

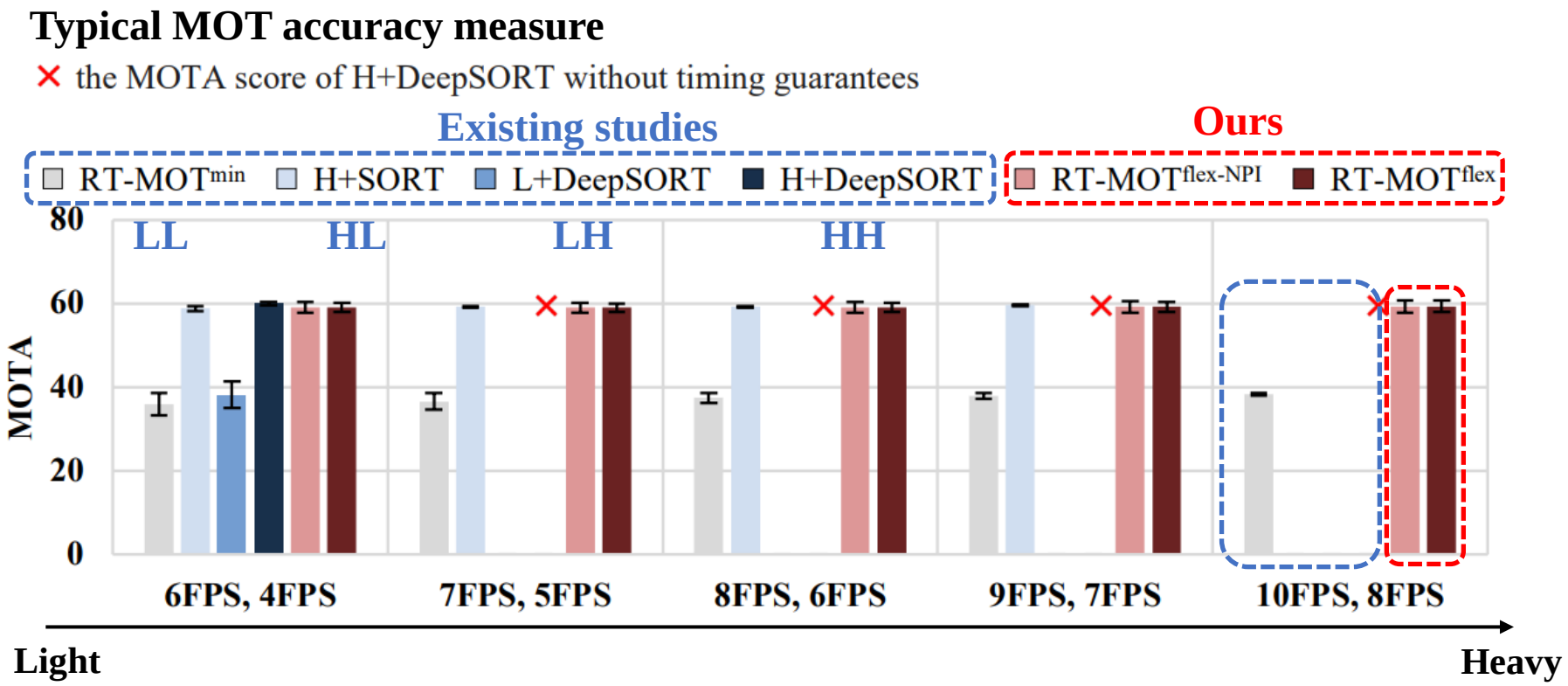
  - **Hardware**: Intel(R) Xeon(R) Silver 4215R CPU @ 3.20GHz, 251.5GB RAM, and NVIDIA V100 GPU
  - **Software**: Ubuntu 18.04.4 with CUDA 10.2, and PyTorch 1.10.2

  - **For Detecting**: YOLOv5 (High confidence: 672×672 / Low confidence: 256×256)
  - **For Tracking**: OSNet (High confidence: DeepSORT/ Low confidence: SORT)

  - **Dataset**: Waymo Open Dataset

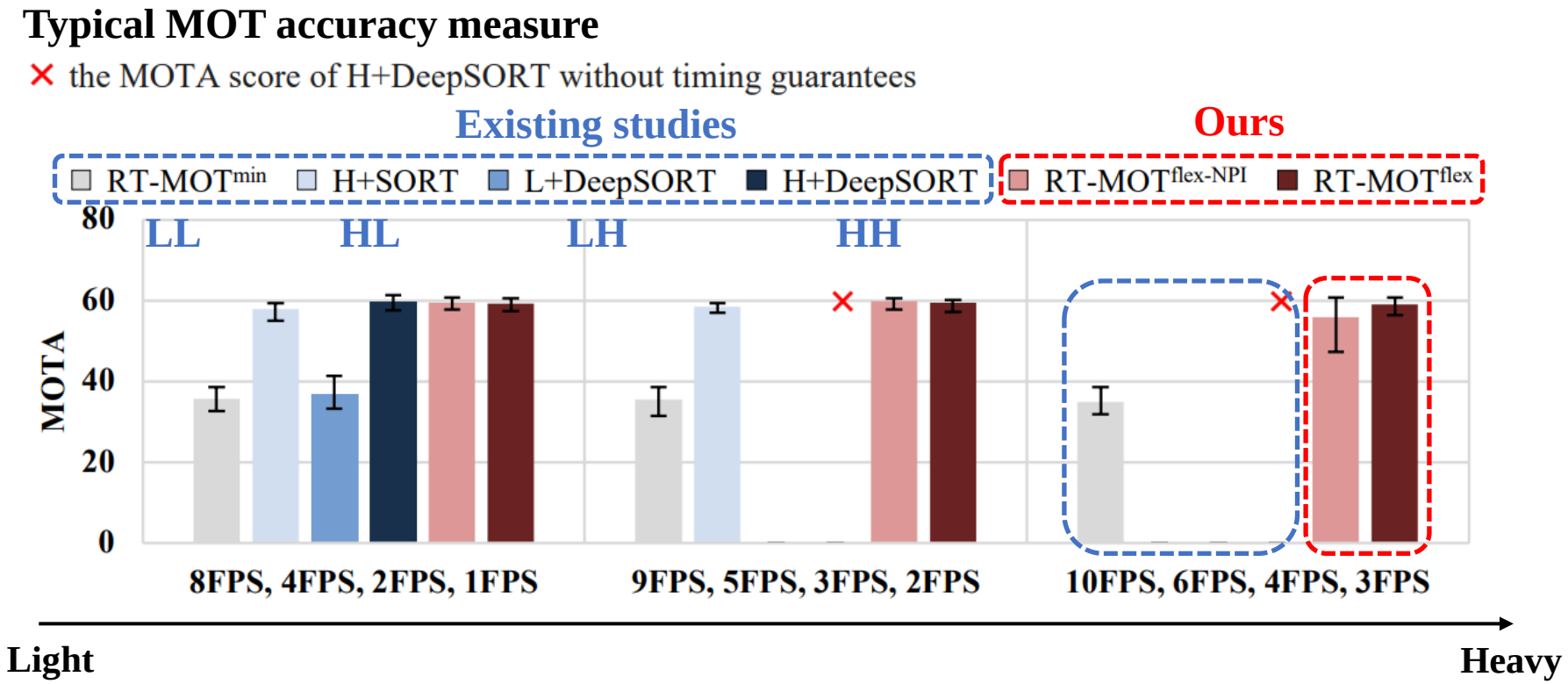
RT-MOT Evaluation

- Under a light workload, existing studies (as well as ours) work well
- Under heavy workload, ours yield timing guarantees with high accuracy



RT-MOT Evaluation

- Under a light workload, existing studies (as well as ours) work well
- Under heavy workload, ours yield timing guarantees with high accuracy



RT-MOT Conclusion

- We propose RT-MOT
  - a new MOT execution and scheduling framework for multiple MOT tasks,
  - which achieves both timing guarantee and accuracy maximization

- RT-MOT implements
  1. **Dynamic tracking-by-detection execution pipeline** to provide multiple choices of a pair of detection and tracking models,
  2. **Confidence estimator** to capture the relation between the object reliability and tracking accuracy, and
  3. **Confidence-aware frame-level scheduler** to select the best model choice for tracking accuracy while providing timing guarantee

논의: 협업가능 연구 주제

Design of *time-predictable* computing systems

+

Timing guarantee and *analysis* of computing systems

- 마감 시간 내 완료 보장, Time-Predictability 등 타이밍에 관한 요건을 만족해야 하는
  - 컴퓨팅 시스템 디자인 및
  - 이에 대한 타이밍 분석 연구
- 그리고 이와 관련된 노하우를 활용한 연구
  - OS/network 등의 스케줄러, 자원관리, 최적화
  - RT for ML 응용, ML for RT 응용
  - 배터리 성능/파워/에너지 고려한 컴퓨팅 시스템 디자인

SAIT 관심 주제

- AI Neural Network 수행 최적화를 위한 complier/runtime
- GPU – CPU DIMM – CXL memory – Memory pool과 같은 tiered memory 환경에서 AI/HPC 응용 수행시 data 두는 시점/위치 최적화