The Price of Schedulability in Cyclic Workloads: The History-vs.-Response-Time-vs.-Accuracy Trade-Off*

Tanya Amert, Ming Yang, Sergey Voronov, Saujas Nandi, Thanh Vu, James H. Anderson, and F. Donelson Smith
Department of Computer Science, University of North Carolina at Chapel Hill

***Abstract*—Autonomous vehicles often employ computer-vision (CV) algorithms that track the movements of pedestrians and other vehicles to maintain safe distances from them. These algorithms are usually expressed as real-time processing graphs that have cycles due to back edges that provide history information. If immediate back history is required, then such a cycle must execute sequentially. Due to this requirement, any graph that contains a cycle with utilization exceeding 1.0 is categorically unschedulable, *i.e.*, bounded graph response times cannot be guaranteed. Unfortunately, such cycles can occur in practice, particularly if conservative execution-time assumptions are made, as befits a safety-critical system. This dilemma can be obviated by allowing older back history, which enables parallelism in cycle execution at the expense of possibly affecting the accuracy of tracking. However, the efficacy of this solution hinges on the resulting history-vs.-response-time-vs.-accuracy trade-off that it exposes. In this paper, this trade-off is explored in depth through both a study of response-time bounds of synthetic task systems and an experimental study conducted using the open-source CARLA autonomous-driving simulator. Somewhat surprisingly, easing away from always requiring immediate back history proved to have only a marginal impact on accuracy, while greatly reducing analytical response-time bounds.**

***Index Terms*—autonomous driving, computer vision, cyber-physical systems, multi-object tracking, real-time systems**

I. Introduction

Semi- and fully autonomous advanced driver-assist systems (ADASs) have become mainstream, as evidenced by systems such as Tesla Autopilot and Cadillac Super Cruise that provide features like adaptive cruise control, automatic lane keeping, *etc.* Such capabilities necessitate the anticipation of dangerous scenarios with enough time for driver or vehicle intervention. Predicting dangerous situations typically entails tracking dynamic objects, such as pedestrians and other vehicles, and using a motion model to extrapolate future positions.

Cameras are cost-effective sensors, so such *multiple-object tracking (MOT)* applications are often image-based, taking a sequence of video frames from a camera as input, and maintaining *tracks* representing the estimated trajectory of each dynamic object over time. An example is shown in Fig. 1, where task τ_1 uses the track produced by task τ_4 during the prior time step to predict the location of an object in the current time step, introducing a *cyclical dependency* on prior results.

*This work was supported by NSF grants CNS 1409175, CNS 1563845, CNS 1717589, CPS 1837337, CPS 2038855, and CPS 2038960, ARO grants W911NF-17-1-0294 and W911NF-20-1-0237, ONR grant N00014-20-1-2698, and funding from General Motors.

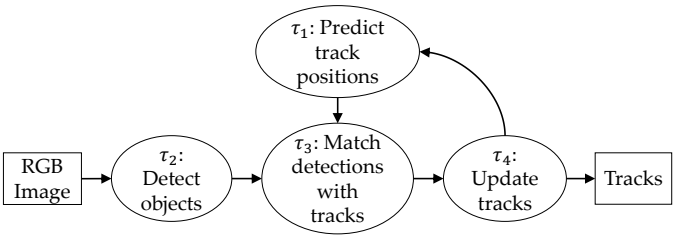


Fig. 1: The tracking-by-detection pipeline.

As this example suggests, tracks corresponding to the *most recent* prior time step are the typical default for prediction in MOT applications. Letting p denote the *maximum prior-history requirement—i.e.*, the maximum difference between the time step in which the prior data is produced and the time step in which it is used—this default corresponds to $p = 1$.

Unfortunately, always insisting on $p = 1$ for every cycle can create a troublesome certification dilemma, because any such cycle must execute sequentially. For example, assuming $p = 1$ for the cycle in Fig. 1, an invocation of task τ_4 requires the results of task τ_3, which requires the results of task τ_1, which requires the results of task τ_4 from the prior time step. Such sequential execution can be problematic because cycles with total utilization[1] exceeding 1.0 can occur in practice, particularly if conservative execution-time assumptions are made, as befits a safety-critical system. For example, such assumptions may account for tracking many more objects than would likely be present, and may reflect the presumption of heavy contention for shared hardware such as caches, memory banks, and buses, as well as accelerators such as graphics processing units (GPUs). The over-utilization caused by such a cycle renders the graph containing it as *unschedulable*, *i.e.*, such a cycle precludes analytically guaranteeing bounded response times for its corresponding graph.

Resolving cycle over-utilization. Assuming $p=1$, any cycle with utilization exceeding 1.0 must have its utilization reduced. There are only two ways to do this (assuming $p=1$): either the cycle's overall execution time must be decreased, or the invocation period of its graph must be increased. However, in any

[1]A cycle's total utilization is given by the total worst-case execution time of all of its nodes (tasks) divided by the corresponding graph's invocation period. Worst-case execution times are determined at design time and can be quite pessimistic in order to ensure that runtime timing violations occur with vanishingly small probability in any system deemed "schedulable."

production system, the former possibility likely would have been applied already in the quest to optimize performance. (Note that simply adding more hardware could violate size, weight, and power (SWaP) constraints that arise in ADASs.)

The latter possibility, increasing a graph's invocation period (*i.e.*, decreasing the invocation rate), has been considered before, albeit in work that does not focus on real-time constraints. This approach equates to using *low-frame-rate tracking*, in which new video frames are available only 5-10 times per second, rather than the standard 20+ times per second. However, as shown by Murray [39], a low frame rate can greatly reduce MOT performance due to larger displacements of the targets being tracked. This reduced performance can be mitigated by using an improved detector [41], [42] or motion model [3]. However, these techniques still completely ignore large portions of the input data; reducing from 20 frames per second (FPS) to 5 FPS ignores 75% of the information potentially available to the CV application!

Recent work by Amert *et al.* [2] on the real-time analysis of graph-based task systems with cyclic dependencies suggests a different way forward: instead of insisting on *sequential* cycle execution, which is the root cause of any over-utilization, allow *parallel* execution instead by permitting the use of slightly older history, *i.e.*, $p > 1$. As shown by Amert *et al.*, allowing $p > 1$ enables the computation of response-time bounds for systems containing cycles with utilization exceeding 1.0. Furthermore, these bounds decrease as p increases, so even for a schedulable system, it may be worthwhile to consider increasing this parameter.

While allowing $p > 1$ may seem heavy-handed, a graph containing a cycle with utilization exceeding 1.0 ***is not schedulable*** if $p = 1$ is assumed. Moreover, recall that p is a *maximum* prior-history requirement, meaning that the oldest data consumed by the ℓ^{th} cycle execution would be from the k^{th} cycle execution, where $k = \ell - p$. At runtime, individual graph nodes would likely execute for far less than their provisioned *worst-case* execution times, so more recent history may be available. Thus, a system with $p > 1$ may have accuracy close to one with $p = 1$ while still being analyzable.

Despite these observations, allowing $p > 1$ is clearly not a solution that comes entirely "for free": as p increases, although response-time bounds may decrease, so too may CV accuracy decrease due to using older history. This accuracy-versus-history trade-off is a key issue in the real-time certification of ADASs of which both CV researchers[2] and automotive designers should be aware, yet it has never been examined in depth. In an attempt to foster such an awareness, we provide here the first-ever detailed study of this trade-off. Furthermore, the analytical benefit of increasing p was not explored in depth in prior work. In this paper, we perform a large-scale study of response-time bounds for graph-based task systems containing cycles.

Schedulability study overview. The response-time analysis provided by Amert *et al.* [2] was evaluated with a single case-study application, but no large-scale study was performed. In this paper, we explore the impact on analytical response-time bounds as p increases for a large set of synthetically generated cycle-containing graph-based task systems, and show that increasing p from 2 to 4 enables average analytical gains of up to 37.6%.

Accuracy study overview. In order to observe the impact of increasing p on the accuracy of a cyclic workload, we consider an MOT system in which pedestrians and other vehicles are tracked via images recorded by a camera attached to a moving vehicle. To explore the history-versus-accuracy trade-off for tracking in isolation, we first assume that all sensors are perfect, *e.g.*, using ground-truth positions of all pedestrians and vehicles in each time step. However, this assumption is not valid for real-world scenarios, so we also evaluate the impact given a CV-based object detector rather than ground-truth data.

We perform our evaluation using CARLA [12], an open-source simulator designed for research on autonomous-driving systems, enabling us to generate a broad range of scenarios, to consider each sensing component independently, and to consider and evaluate potential modifications to the vehicle's behavior based on tracking results. Our results show that allowing p to increase slightly has only a minor impact on tracking accuracy, whereas low-frame-rate tracking (effectively enforcing a much higher p, even if more recent results are available) suffers greatly reduced tracking accuracy. When using a CV-based object detector, enabling $p > 1$ resulted in up to 13.35% higher precision and 8.74% higher accuracy than using half the original frame rate, as long as the most recent results were available approximately 80% of the time.

Finally, we present the results of a case study in which we executed our detector-based MOT application running alongside other real-time tasks used in automotive applications. We measured the observed distribution of historical results, and found that even though the most recent results were only available approximately 65% of the time, the tracking accuracy was comparable to sequential execution, and there was only a 5.1% drop in precision.

Organization. The remainder of this paper is organized as follows. In Sec. II, we discuss the real-time scheduling implications of varying parallelism and the computation of response-time bounds. In Sec. III, we discuss the results of our experimental evaluation of the history-versus-response-time trade-off. Next, we provide detailed background on the MOT pipeline in Sec. IV before giving an overview of our evaluation of the history-versus-accuracy trade-off in Sec. V. In Sec. VI we discuss the results of our accuracy evaluation, both when ground-truth detections are used and in the presence of a CV-based object detector. We present our case study in Sec. VII and conclude in Sec. VIII.

[2]The fact that this trade-off has not been considered before by CV researchers is not surprising, given its roots in real-time schedulability concerns.

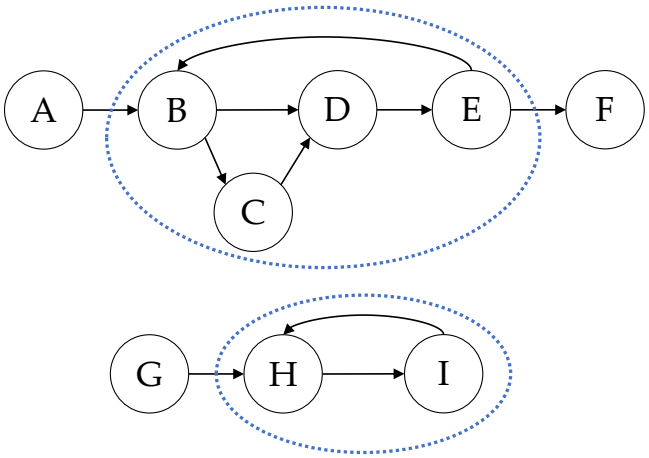


Fig. 2: A set of nine nodes in two graphs, each containing a cycle (indicated by the dashed ovals).

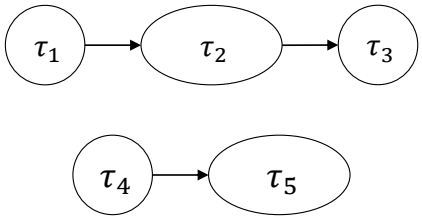


Fig. 3: The task set from Fig. 2 after converting each cycle to a supernode. Nodes have been relabeled to correspond to individual sporadic tasks.

II. Real-Time Systems Background

In this section, we provide necessary background on the real-time scheduling and analysis of graphs containing cycles.

A. Real-Time Graph Scheduling

Much prior work on response-time analysis under global schedulers [11], [18], [31], [32] has assumed the sequential sporadic task model, in which invocations of the same task execute sequentially, *i.e.*, no two invocations of task τ_1 in Fig. 1 may execute concurrently. In contrast, full intra-task parallelism, in which any number of invocations of a given task may execute concurrently, has been shown to enable much smaller response-time bounds [17]. However, full parallelism requires that the utilization of each cycle is low.

Transforming a cyclic graph to a DAG. Prior work on response-time analysis for graph-based workloads primarily considers DAGs. Therefore, the nodes comprising any cycle in a graph must be replaced by a single "supernode" [52]. We demonstrate scheduling a cycle-containing graph-based task system using a continuing example.

Ex. 1. Two graph-based applications are depicted in Fig. 2. In the top graph, nodes $\{B, C, D, E\}$ form a cycle. As shown in Fig. 3, this cycle must be converted to a supernode (τ_2). Similarly, nodes $\{H, I\}$ are merged into τ_5 in Fig. 3. ◊

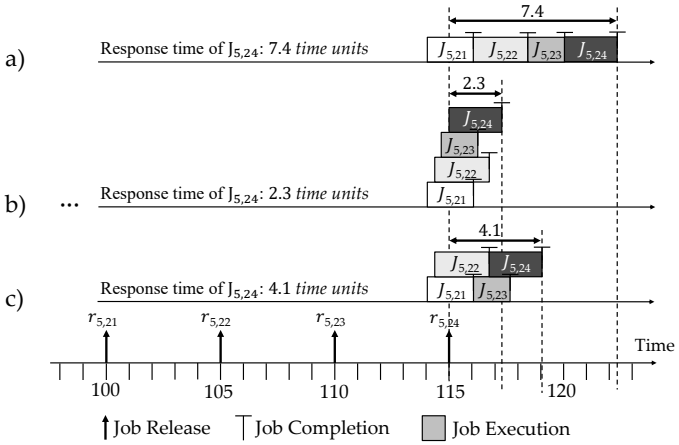


Fig. 4: Scheduling repercussions of the degree of intra-task parallelism, including a) sequential execution, b) fully parallel execution, and c) restricted parallelism. Successive jobs are shaded progressively darker. Assume there are additional tasks in the system beyond those in Fig. 3, and that the depicted jobs are scheduled alongside other jobs, which are not shown.

Scheduling DAG-based task sets. After converting a graph to a DAG, prior work can be leveraged to further transform the DAG into a set of independent sporadic tasks [35], [52], [53]; a summary of this process is given by Amert *et al.* [2].

Scheduling of these tasks depends on the choice of intra-task parallelism.

Ex. 1 (cont'd). Fig. 4 depicts possible schedules for invocations (called *jobs*) of the supernode τ_5 on a platform with four CPUs. The kth job of τ_5, $J_{5,k}$, is released at time $r_{5,k}$.

In schedule (a), the jobs execute sequentially. Due to jobs of other tasks (not shown), $J_{5,21}$ is released at time 100, but not scheduled until time 114. This delay impacts the subsequent jobs; $J_{5,24}$ has a response time of 7.4. However, the $p = 2$ requirement is met, *i.e.*, $J_{5,21}$ completes before $J_{5,23}$ begins.

Schedule (b) shows the result of fully parallel execution. The response time of $J_{5,24}$ is reduced to 2.3 time units. ◊

However, unrestricted intra-task parallelism can violate the dependencies required by back edges in cycles.

Ex. 1 (cont'd). Assume that task τ_5, a supernode, was created from a cycle with worst-case history age $p = 2$. Thus, job $J_{5,23}$ requires output from one of jobs $J_{5,21}$ and $J_{5,22}$. However, in schedule (b) of Fig. 4, jobs $J_{5,21}$, $J_{5,22}$, and $J_{5,23}$ all execute concurrently, violating this precedence constraint. ◊

Unfortunately, sequential execution can result in an unschedulable task system. Under the sporadic task model, a task system τ is comprised of tasks τ_i, specified as $\tau_i = (\Phi_i, T_i, C_i)$, where Φ_i is the release time of the first job of τ_i, T_i is the minimum separation between job releases, and C_i is the worst-case execution time (WCET) of any job of τ_i. The maximum WCET of any task is given by C_{max}. The utilization of task τ_i is given by $u_i = C_i/T_i$, and the total system utilization is $U = \sum_i u_i$.

Task	C_i	T_i	P_i	u_i
τ_1	4	10	m	0.4
τ_2	12	10	2	1.2
τ_3	2	10	m	0.2
τ_4	1	5	m	0.2
τ_5	4	5	1	0.8

TABLE I: Task parameters for the task system in Ex. 2, assuming a platform with $m = 3$ CPUs.

Ex. 1 (cont'd). If jobs execute sequentially as in Fig. 4(a), response times can be unbounded for τ_5 if $u_5 > 1.0$. ◊

The *restricted parallelism* sporadic (*rp-sporadic*) task model introduced by Amert *et al.* [2] adds a per-task parallelism value P_i, which specifies the maximum number of jobs of τ_i that may execute concurrently; any $P_i \leq p$ guarantees that prior-history requirements are met. Note that this task model generalizes both sequential ($P_i = 1$) and fully parallel ($P_i = m$, where m is the number of CPUs) execution.

Ex. 1 (cont'd). Restricted intra-task parallelism ($P_5 = 2$) is shown in schedule (c) of Fig. 4. The response time of $J_{5,24}$ is increased to 4.1, but history requirements are respected. ◊

B. Bounding Response Times

Formally, the response time $R_{i,k}$ of a job $J_{i,k}$ is the time between its release, $r_{i,k}$, and its finish time, $f_{i,k}$: $R_{i,k} = f_{i,k} - r_{i,k}$. The response times of rp-sporadic tasks can be bounded using analysis provided by Amert *et al.* [2]. As in this prior work, we assume global earliest-deadline first (G-EDF) scheduling on a system with m CPUs, and that access to any accelerators (*e.g.*, GPUs) is non-preemptive and managed by locking protocols. We let B_{max} be the duration of the longest non-preemptive accelerator access, and assume that per-task WCETs have been inflated to account for locking-protocol-related blocking.

Feasibility conditions. In addition to the system utilization constraint of $U \leq m$, for an rp-sporadic task system to have bounded response times, the utilization of any cycle must be at most P_i. Thus, the second feasibility condition is $\forall i : u_i \leq P_i$.

Ex. 2. Consider again the task set in Figs. 2 and 3. Possible parameters of these tasks are given in Table I. For this example, assume that $m = 3$. The first feasibility condition is satisfied, as $U = 2.8$. Additionally, note that for all tasks, $P_i \geq u_i$, so the second feasibility condition is also satisfied. Furthermore, as $P_i \leq 2$ (again assume $p = 2$ for both cycles), the precedence constraints are not violated. ◊

Computing response-time bounds. In this paper, we explore the history-versus-response-time trade-off through an experimental evaluation. For this study, we make use of the closed-form bound provided by Amert *et al.* This bound relies on the summed utilizations and WCETs of the tasks with restricted parallelism. This required definition, as well as the bound, are copied below from [2].

Def. 1. (Def. 5 in [2]) *Call a task τ_i* p-restricted *(parallelism-restricted) if $P_i < m$, and* non-p-restricted *if $P_i \geq m$. Also, let*

$$U^b_{res} = \sum_{\substack{b \text{ largest values} \\ \tau_i \text{ is p-restricted}}} u_i \quad \text{and} \quad C^b_{res} = \sum_{\substack{b \text{ largest values} \\ \tau_i \text{ is p-restricted}}} C_i,$$

and let $U_{res} = U^n_{res}$ and $C_{res} = C^n_{res}$.

Corollary 1. (Cor. 1 in [2]) *The response time of any task $\tau_i \in \tau$ is bounded by $x + T_i + C_i$, where*

$$x = \frac{(m-1)C_{max} + B_{max} + 2C_{res}}{m - U_{res}}. \tag{1}$$

Furthermore, if there exists $P_{min} \geq 1$ such that for every p-restricted task τ_i, $P_i \geq P_{min}$, then U_{res} and C_{res} in (1) can be replaced with U^ℓ_{res} and C^ℓ_{res}, where $\ell = \lfloor (m-1)/P_{min} \rfloor$.

Cor. 1 enables the calculation of an upper-bound on the response time of any task in the system. To bound the end-to-end response time of a DAG, we take the maximum sum of the response times of each task along any path in the DAG.[3]

Ex. 2 (cont'd). Assume a maximum non-preemptive GPU access duration of $B_{max} = 2$ time units. For the system described in Table I, $C_{max} = 12$, $C^\ell_{res} = 16$, $U^\ell_{res} = 2.0$. Cor. 1 gives a value for x of 58. Thus, the end-to-end response time of the graphs are $72 + 80 + 70 = 222$ and $64 + 67 = 131$ time units, respectively. ◊

Trading off history and response-time bounds. To compare bounds between graphs with different periods, we instead refer to the maximum *relative tardiness* of a graph. Tardiness is defined as the amount by which a job misses its deadline: $\max\{0, f_{i,k} - d_{i,k}\}$, where the absolute deadline $d_{i,k}$ is assumed to be implicit (*i.e.*, $d_{i,k} = r_{i,k} + T_i$). Note that tardiness cannot be negative. Given a response-time bound R for a graph, the relative tardiness is given by $(R - T_i)/T_i$ (we assume every node in a graph shares the same period).

Ex. 2 (cont'd). The graph comprised of tasks τ_1, τ_2, and τ_3 has a period of 10 time units and a response-time bound of $R = 222$ time units. Thus, its maximum relative tardiness is 21.2. Similarly, the graph comprised of tasks τ_4 and τ_5 has a period of 5 time units and a response-time bound of $R = 131$ time units, so its maximum relative tardiness is 25.2. ◊

Although the per-task parallelism P_i is constrained from below by the utilization of a supernode, it can be as high as p, albeit with a potential loss of algorithmic accuracy. Increasing P_i can increase P_{min} and thus decrease ℓ, leading to drastic reductions in analytical response-time bounds.

Ex. 2 (cont'd). If we instead set $P_5 = 2$, ℓ decreases to 1, reducing C^ℓ_{res} and U^ℓ_{res} to 12 and 1.2, respectively. As a result, $x = 27.8$, so the graphs' end-to-end response-time bounds are reduced to 131.3 and 70.6 time units, respectively. Therefore, the maximum relative tardiness bounds are reduced to 12.1 and 13.1, respectively, a reduction of over 42%. ◊

[3]Formally, there should be a single source and sink node; if multiple such nodes exist, a single *virtual* node can be added with $C_i = 0$.

This example demonstrates that allowing older history to be used can reduce the analytical bounds on tardiness (and thus response times) for a task system. In the next section, we provide an in-depth evaluation of this history-versus-response-time trade-off.

III. Evaluating the History-versus-Response-Time Trade-Off

Given the intuition provided by Ex. 2, we now explore the impact of changing P_{min} on the maximum end-to-end relative tardiness of a graph-based task system.

A. Experimental Setup

We consider a platform with $m = 16$ CPUs. Recall from the response-time bound calculation in Cor. 1 that B_{max} is independent of P_{min}. Thus, we did not consider GPU usage in our experiments (so $B_{max} = 0$). Instead, we focused on the effects of changing U_{res} and C_{res} by increasing P_{min} past the minimum possible value given by the per-task feasibility condition $u_i \leq P_i$.

In this study, we generated $200,000$ graph-based task systems, with system utilizations in the range $[2.5, 16]$. To generate a task system, we first chose a target system utilization value, and selected per-task utilizations from a given distribution (described below) until an additional task would cause total utilization to exceed m. Then, tasks were randomly assigned to graphs such that four to eight tasks were assigned to each graph (once fewer than four tasks remained, the rest were assigned to a final graph). Finally, edges were selected with probability 0.3, according to the Erdős-Rényi graph generation model [16], which is commonly used to generate random graphs.[4]

We selected per-task utilizations from two distributions: *uniform* and *exponential*. For the uniform distribution, the utilization of each task was independently chosen uniformly from $[0, 1.5)$. Thus, approximately one-third of tasks generated using this distribution required $P_i \geq 2$. When using the exponential distribution, each task had utilization chosen independently from an exponential distribution with mean 0.6. With both distributions, any task with $u_i \leq 1$ was assumed to have $P_i = m$; otherwise, P_i was set to $\lceil u_i \rceil$. Periods were chosen per-graph uniformly from $[10, 100]$.

B. Results

After computing the maximum relative tardiness for each task system, we grouped the results into system-utilization buckets, with bucket sizes of 1.0 for low system utilizations to 0.5 for high system utilizations, and report the average of the maximum relative tardiness values for each utilization bucket. The results of our tardiness study are shown in Fig. 5 and Fig. 6 for uniform and exponential per-task utilizations, respectively. Each figure contains three curves, one for each of three values of P_{min}. In this context, P_{min} corresponds to the minimum P_i of any task in the system. To ensure that

[4]Each of the possible edges in the graph was independently added with probability 0.3. Note that we do not require a graph to be connected.

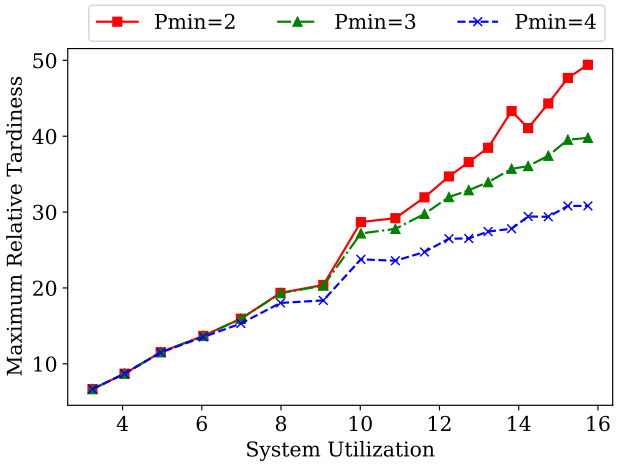


Fig. 5: Maximum relative tardiness results for uniform per-task utilizations.

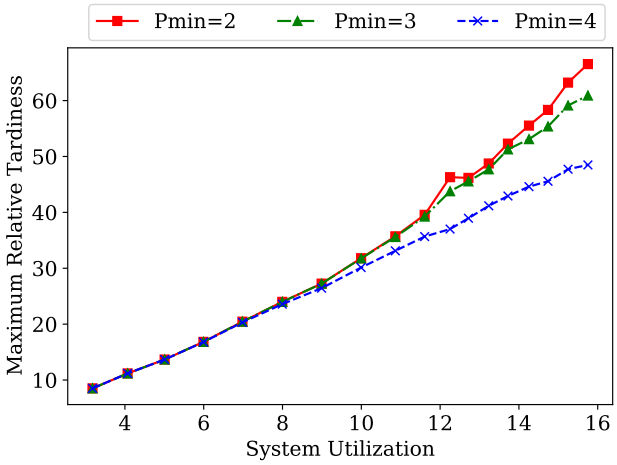


Fig. 6: Maximum relative tardiness for exponential per-task utilizations.

$P_{min} = 2$ was valid for each task system, we regenerated the per-task utilizations (as described in Sec. III-A) if no task had $P_i = 2$.

Increasing P_{min} corresponds to increasing the value of P_i for a subset of supernodes in the system, thus allowing more parallelism through the use of (possibly) older history. Given Cor. 1, this should decrease the relative tardiness for the task system. Thus intuition is borne out in both figures. Allowing $P_{min} = 3$ instead of $P_{min} = 2$ led to a 19.5% reduction in relative tardiness for uniform tasks, and a 8.5% reduction for exponential tasks. Allowing instead $P_{min} = 4$ greatly increased the reduction in relative tardiness, by 37.6% and 27.1% for uniform and exponential tasks, respectively.

These results demonstrate the huge analytical improvement that can be gained by allowing the use of older historical information. However, this gain comes at a price; the accuracy side

of the history-versus-response-time-versus-accuracy trade-off will be explored next, after some necessary background.

IV. MOT Background

In this section, we provide background on a Tracking-By-Detection multi-object tracking pipeline.

A. MOT via Tracking-By-Detection

MOT tracks an unknown number of objects, or *targets*, through a scene. A *track* is a sequence of estimated positions and sizes (as bounding boxes) of a target over time. A track is a model of a target's *trajectory*, *i.e.*, the sequence of its actual real-world positions. Time is measured by camera frames.

Tracking-by-detection (TBD) is a common approach to MOT. This pipeline is illustrated in Fig. 1. The output from frame t is the set of tracks after frame t. The input to frame t is an RGB image and the set of tracks from frame $t-1$. We now describe each step, including a few representative implementations.

Predicting track positions. Given a set of tracks from frame $t-1$, a *motion model* is used to predict the position (represented as a bounding box) of each tracked target in frame t.

Ex. 3. Fig. 7 depicts the results of each of the four TBD steps (the order matches the task indices in Fig. 1) for a given frame. Dashed boxes in Fig. 7a represent predictions of vehicles' positions in the current frame. ◊

In order to predict the new position of a target, a model of its motion must be used to extrapolate from the existing track. The simplest motion model assumes constant velocity: the two most recent track positions are used to linearly extrapolate the next. More advanced motion models use curvilinear extrapolation [48], particle filtering [6], [27], or optical flow [28].

Multi-object detection. The goal of multi-object detection is to identify the positions of all targets in an RGB image. The number of targets is not known *a priori*.

Ex. 3 (cont'd). The detection step outputs a bounding box for each detected vehicle, as in Fig. 7b. ◊

Multi-object detection is typically performed in two stages: features are selected from regions in the image, and then each feature is classified to determine if it is part of a relevant bounding box. A traditional approach is feature selection via histogram of oriented gradients (HOG) [9] followed by classification using linear support vector machines (SVMs). Recently, deep convolutional neural networks have been used instead [21], [24], [44].

Matching detections to tracks. Given a set of detected bounding boxes and predictions of new track positions, the percentage overlap is compared for all detection-prediction rectangle pairs. The Hungarian method (also known as Munkres' algorithm) can be used to quickly match detections to predictions [36], [49]. The overlap of two rectangles is computed using the *intersection-over-union* measure (IOU) [45], also known as the Jaccard index. The IoU (a scalar) is the ratio of the size of the intersection to the size of the union of two rectangles within an image. The Hungarian algorithm chooses an assignment of detections to predictions that maximizes the IoU of the selected pairs. The output of this step is a set of detection-prediction assignments, as well as the lists of detections and predictions that are unmatched.

Ex. 3 (cont'd). Matching between detections (solid boxes) and tracking predictions (dashed boxes) are shown in Fig. 7c. ◊

Updating the tracks. For each prediction that is matched with a detection, the corresponding track is updated based on the detected position. Depending on the motion model, the model is also updated based on the new position. If a track has enough consecutive unmatched predictions, then it can be deleted. Unmatched detections potentially correspond to newly visible objects; for each unmatched detection, a new track is created. (More complex filtering can be done to handle noisy detections, if necessary.)

Ex. 3 (cont'd). Tracks corresponding to the two matched predictions are updated to contain a new position based on the detection, as shown in Fig. 7d. ◊

B. MOT from a Moving Vehicle

The pipeline described in Sec. IV-A assumes a stationary camera. For a vehicle-mounted camera, it is necessary to account for *ego-motion*, *i.e.*, the motion of the camera itself.

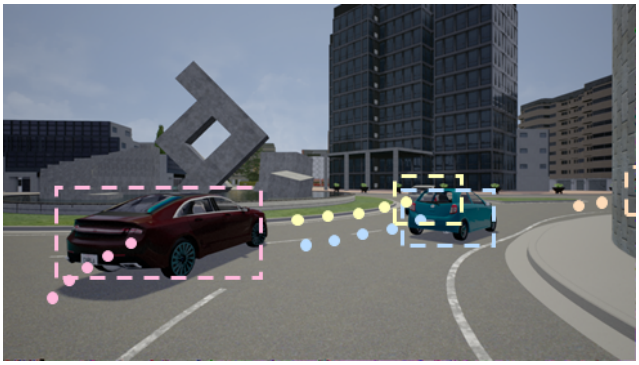
Typical approaches to ego-motion estimation fit into two categories: *simultaneous localization and mapping (SLAM)* methods [7], [14], [37], [38] and *structure-from-motion (SfM)* methods [20], [22], [43], [47]. Both approaches leverage the overlap of static scene content (*e.g.*, road signs, buildings) between frames, as well as the motion of such content, to estimate the movement of the camera. SLAM methods generally assume frames are temporally sequential, whereas SfM methods allow frames to be processed in any order.

In addition to knowing where the camera moves within the world, it is also necessary to determine each target's real-world position relative to the camera. *Stereo-estimation* methods [4], [13], [23], [33], [46], [51], [54], [55] determine the depth (distance to the 3D scene point) for each pixel by examining corresponding camera poses for a pair of frames, using either two cameras mounted on the car, or a single moving camera at two different points in time.

C. Input for MOT

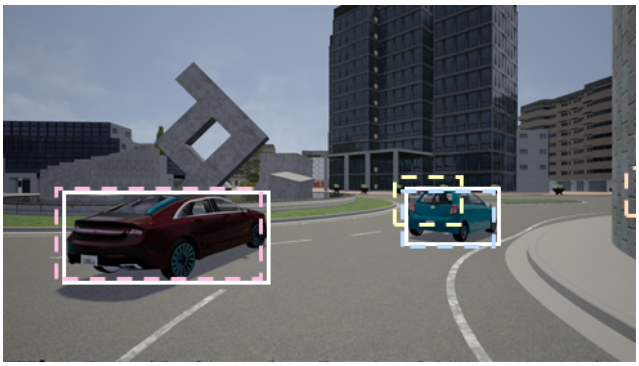
We can now list the required inputs for the tracking component of a TBD-based MOT application in which the camera is moving. First, to determine the positions of targets relative to the camera, an RGB image is needed along with a distance value for each pixel in that image. Then, the detector provides a set of bounding boxes corresponding to 2D rectangles in the RGB image. Finally, ego-motion estimation provides the relative movement of the camera in the 3D world.

In this paper, we use a simulator that provides the ground-truth position of the camera. Therefore, we do not need to perform ego-motion estimation. The primary input we consider is thus the bounding boxes of the detections. When discussing the



(a) Each track (the output of processing the previous frame) is extrapolated to the current frame based on the choice of motion model. In this example, four tracks (indicated by dots) are used to predict the positions of four vehicles in the current frame; three vehicles are visible, and one is out of view to the right. The predicted positions for this frame are depicted as dashed rectangles.

(b) Given an RGB image, the detector outputs a bounding box for each detected object. In this example, a vehicle on the right occludes another, so only two objects are detected. As the detector has no knowledge of the mapping from bounding boxes to vehicles, detected bounding boxes are shown as solid white rectangles, rather than colored based on the targets being tracked, as in inset (a).



(c) Matching occurs between the predicted positions of each track and the detected bounding boxes. In this example, two of the four tracks are matched to a detection.

(d) Tracks matched to a detection are updated with a new position based on that detection. Unmatched tracks are either deleted (if unmatched long enough) or updated with the predicted position.

Fig. 7: The output of each step of the TBD pipeline from Fig. 1 for a single camera frame.

results of our evaluation in Sec. VI, we consider first a system with ground-truth detections in order to evaluate tracking in isolation, and then a system with CV-based detections. In the next section, we describe our experimental evaluation of this history-versus-accuracy trade-off in full.

V. Evaluating the History-versus-Accuracy Trade-Off

We now describe the experiments we performed to evaluate the trade-off between history and accuracy. We begin by giving an overview of our experimental setup and traffic scenario selection, and then discuss how we varied the age of historical data provided to the MOT application.

A. Experimental Setup

We conducted our evaluation using *CARLA* [12], an open-source urban-driving simulator. *CARLA* uses Unreal Engine 4 [15] to produce photo-realistic scenes combined with accurate physical models of automobile dynamics. *CARLA* is a client-server system. The urban environment and the interactions of all vehicles and pedestrians with it are simulated on the server. The client sends parameters for steering, acceleration, and braking to the server, and is controlled manually or via an *agent* that implements the perception, planning, and control elements for driving; in our experiments, the vehicle was controlled manually. All sensor data is provided by the server, including physically based graphics renderings of camera frames. Additionally, the server provides ground-truth data needed for evaluation, such as the location and orientation of the camera and each vehicle and pedestrian in the scene.

We evaluated the impact of increasing p in a broad range of scenarios generated from *CARLA*. These scenarios need to be challenging driving situations that require highly accurate tracking. The *CARLA Challenge* provides scenarios that are selected from the NHTSA (National Highway Traffic Safety Administration) pre-crash typology [40], which provides scenarios that are identified as common pre-crash scenarios of all police-reported crashes. From these scenarios, we selected the list below, which heavily rely on tracking and prediction, as our focus. We modified each scenario by adding additional vehicles and pedestrians to make the tracking task more

challenging. In the descriptions that follow, "ego-vehicle" refers to the vehicle that navigates the scenario.

- **Scenario 1:** *Obstacle avoidance with prior action.* As the ego-vehicle turns right at a red light, an unexpected obstacle (a cyclist) crosses into the road. The ego-vehicle must perform an emergency brake or an avoidance maneuver.
- **Scenario 2:** *Right turn at an intersection with crossing traffic.* The ego-vehicle must turn right on red at an intersection and in the presence of crossing traffic. In this scenario, the ego-vehicle must track all crossing vehicles, yielding to avoid collisions.
- **Scenario 3:** *Crossing traffic running a red light at an intersection.* As the ego-vehicle enters an intersection going straight, a vehicle runs a red light from the right. In this scenario, the ego-vehicle must perform a collision avoidance maneuver.
- **Scenario 4:** *Unprotected left turn at an intersection with oncoming traffic.* The ego-vehicle must perform an unprotected left turn at an intersection, yielding to oncoming traffic.
- **Scenario 5:** *Lane changing to pass a slow-moving leading vehicle.* While driving on the highway, the vehicle in front of the ego-vehicle rapidly decelerates. The ego-vehicle must change lanes to avoid a collision, yielding to traffic in the next lane.

The five scenarios are depicted in Fig. 8. Scenarios 1-4 feature city driving; Scenario 5 involves highway speeds. In our experiments, each scene is populated with additional vehicles and pedestrians that obey all traffic laws (*e.g.*, additional pedestrians do not enter the road, additional vehicles and cyclists obey stop lights and lane markings). Scenario 1 features three vehicles and twelve pedestrians. Scenarios 2-4 have six vehicles, and Scenarios 2 and 4 also have four pedestrians. Scenario 5 contains eleven vehicles.

B. Varying the Age of History

For our experiments, we implemented the TBD pipeline in Fig. 1. As described in Sec. I, this graph contains a cycle comprised of tasks τ_1, τ_3, and τ_4. The prior-history requirement p for the back edge from τ_4 to τ_1 is what we seek to vary; increasing p means that the track prediction step (τ_1) may use less-recently updated tracks to make predictions.

By definition, p is the *maximum* difference in time steps between the completion of an invocation of τ_4 and when those results are used by τ_1. However, more recent results can be used, if available. In our evaluation, we represent the distribution of available prior results using a *probability mass function* (PMF), which we represent as a tuple.

To measure the impact of varying p in our experiments, we executed the code sequentially, and for each invocation of τ_1, we chose the prior history to use based on a random number sampled from the PMF. For example, for a PMF represented as $(0.8, 0.2)$, we selected one frame prior with probability 0.8, and two frames prior with probability 0.2.

We evaluated eight PMFs, listed in Table II, chosen to answer four questions:

	PMF (represented as a tuple)
PMF 1	$(0.9, 0.09, 0.009, 0.001)$
PMF 2	$(0.8, 0.2)$
PMF 3	$(0.8, 0.02, 0.02, 0.16)$
PMF 4	$(0.5, 0.4, 0.1)$
PMF H_p	$\left(0, \overset{p-1}{\ldots}, 0, 1\right)$

TABLE II: Probability mass functions (PMFs) corresponding to available history results. The PMFs are described as tuples: (x, y) indicates that the result of one and two frames prior are available with probabilities x and y, respectively. In *PMF* H_p, $0, \overset{p-1}{\ldots}, 0$ denotes a sequence of $p-1$ 0's, where p ranges over $(1, ..., 4)$. Note that the values of each PMF sum to 1.0.

- **Q1** What if the most recent data are sometimes unavailable?
- **Q2** How much of an impact does the worst-case age have if the most recent data are usually available?
- **Q3** Is it better to have a higher chance of more recent data, or a lower worst-case age?
- **Q4** How does the average case differ from the worst case?

Comparing *PMFs* 1 and H_1 should answer question Q1. We can answer question Q2 by comparing *PMFs* 2 and 3. For question Q3 we compare *PMFs* 3 and 4. Finally, we compare *PMFs* 1-4 to the corresponding worst-case *PMF* H_p (*e.g.*, *PMFs* 4 and H_3) to answer question Q4.

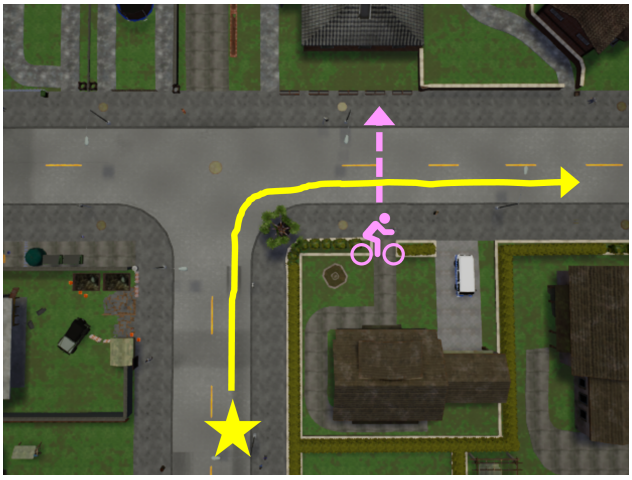
VI. Results: History-versus-Accuracy

We first consider tracking in isolation, *i.e.*, in the presence of perfect sensors. For each frame of the video, we provide the ground-truth 3D motion of the camera (representing perfect ego-motion estimation), ground-truth 2D bounding boxes (as if from a perfect detector), and the 3D distance to each target within the scene (corresponding to perfect stereo estimation). (In Sec. VI-C, we remove the assumption of a perfect detector.)

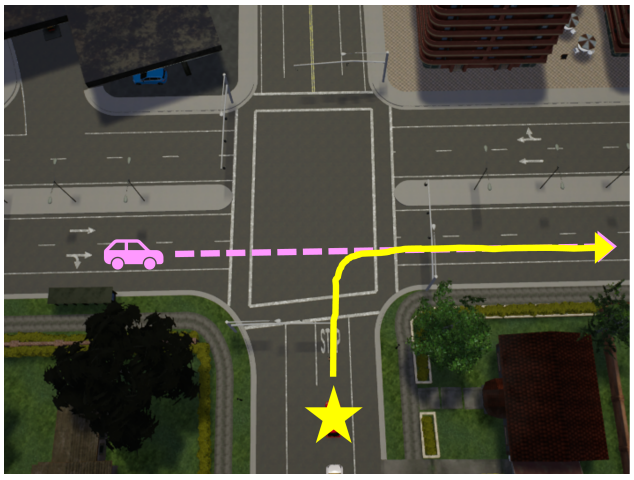
A. Evaluation Metrics for MOT

In this section, we overview of the standard metrics used to evaluate MOT applications [49]. We consider first the metrics defined for each frame and then the high-level MOT metrics.

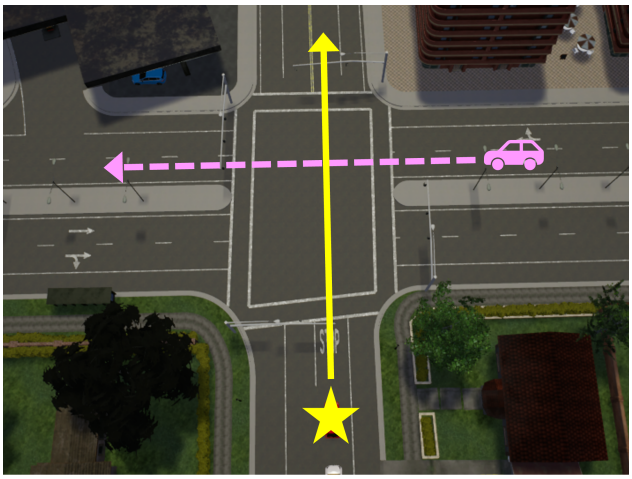
Per-frame metrics. A track represents not only where a target is believed to have been, but the ability to predict where it will be in future frames. Thus, a track for which the prediction is matched to a detection is considered a *true positive*, and we let TP_t be the number of such matches for frame t. An unmatched prediction is a hypothesis that does not correspond to any detected target (false positive), and an unmatched detection corresponds to a target for which there is no such hypothesis (false negative). These are represented by FP_t and FN_t, respectively. The ground-truth number of targets present in frame t is represented by GT_t. In the best case, $TP_t = GT_t$ and $FN_t = FP_t = 0$.



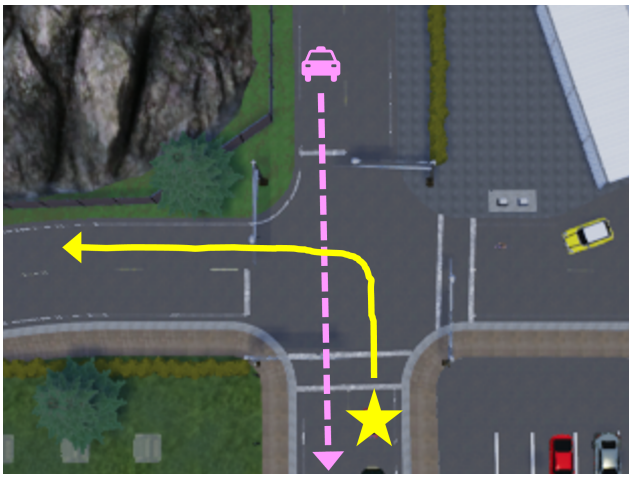
(a) Scenario 1: Obstacle avoidance with prior action.



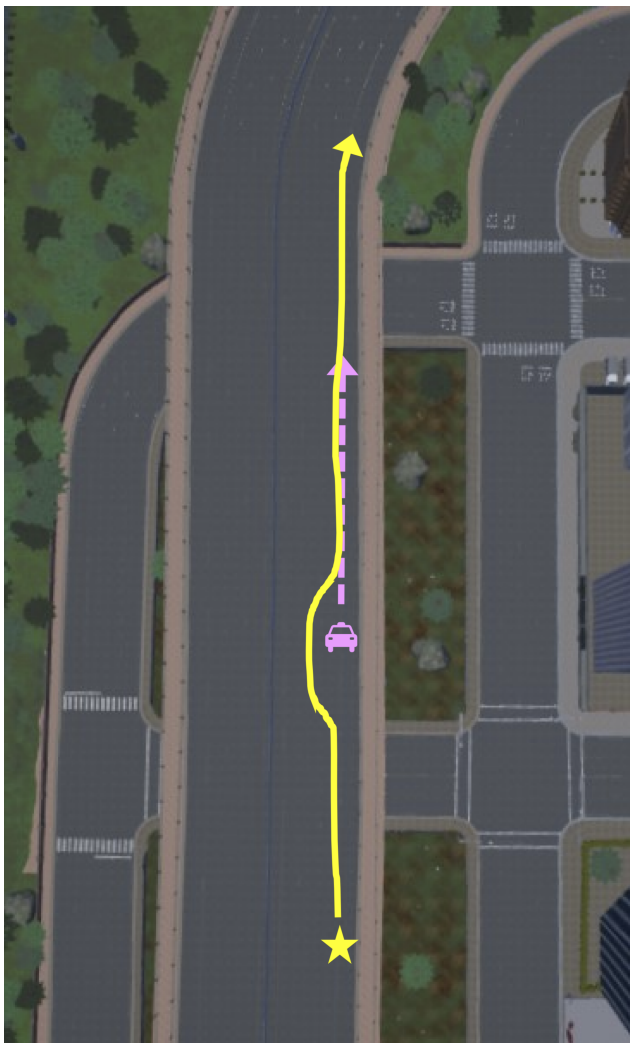
(b) Scenario 2: Right turn at an intersection with crossing traffic.



(c) Scenario 3: Crossing traffic running a red light at an intersection.



(d) Scenario 4: Unprotected left turn at an intersection with oncoming traffic.



(e) Scenario 5: Lane changing to pass a slow-moving leading vehicle.

Fig. 8: The five scenarios we explore. The start position and path of the ego-vehicle are indicated by a yellow star and solid line, respectively. The cyclist/vehicle the ego-vehicle must avoid are indicated with a pink dashed path and appropriate icon; additional vehicles and pedestrians are not shown.

Overall metrics. Several metrics are typically used in the CV literature to evaluate a tracker's performance holistically. In this paper, we focus on two of them: *A-MOTA* and *MOTP*.

The *Average Multiple-Object Tracking Accuracy (A-MOTA)* metric combines information about the false positives, and false negatives:

$$\textit{A-MOTA} = 1 - \frac{\sum_t (FN_t + FP_t)}{\sum_t GT_t}.$$

The other common overall metric for MOT applications is the *Multiple-Object Tracking Precision (MOTP)*, given as a ratio of the distances between ground-truth object positions and predicted track positions and the number of matches made, summed over all objects and all frames:

$$\textit{MOTP} = \frac{\sum_{t,i} d_{t,i}}{\sum_t c_t}.$$

In this expression, $d_{t,i}$ is the IOU of the bounding boxes of object i and its predicted track position at frame t, and c_t is the number of detection-track matches found at frame t. For an ideal tracking system, *MOTP* $= 1.0$ (*i.e.*, each detection-track match has perfect overlap).

B. Perfect Sensing

We now describe the results of tracking both vehicles and pedestrians in the scenarios listed in Sec. V-A, sampling from the PMFs in Table II. Each scenario lasted 300 camera frames. We evaluated the accuracy and precision of tracking, comparing the results for each PMF with those of *PMF* H_1. For each scenario, we collected RGB images generated by a single front-facing camera, ground-truth bounding boxes of all vehicles and pedestrians that were captured by the camera, and the ground-truth motion of the camera itself.

The *A-MOTA* and *MOTP* for each scenario and PMF are reported in Table III for vehicle tracking and Table IV for pedestrian tracking (note that Scenarios 3 and 5 had no pedestrians). For *PMFs* 1-4, we repeated the scenario 100 times, and report the average *A-MOTA* and *MOTP* results. Given these data, we can now answer the four questions posed in Sec. V-B.

Q1: What if the most recent data are sometimes unavailable? To explore this question, we compare the results of *PMFs* 1 and H_1. As *PMF* 1 has the most recent data available with probability 0.9, we expect that the accuracy and precision will be comparable to *PMF* H_1, for which the most recent data are always available.

Comparing the two columns in Tables III and IV, we see that *PMF* 1 has an *A-MOTA* score within 0.62% and 0.90% of that of *PMF* H_1 across all scenarios for vehicles and pedestrians, respectively. Similarly, the *MOTP* score for *PMF* 1 is within 2.07% and 1.61% of that of *PMF* H_1 for vehicles and pedestrians, respectively.

Q2: How much of an impact does the worst-case age have if the most recent data are usually available? This question can be answered by comparing *PMFs* 2 and 3. For both distributions, the most recent data are available with probability 0.8. However, *PMF* 3 represents a bimodal distribution, which may result if tasks become greatly delayed; its worst-case age is four frames, which occurs with probability 0.16. For *PMF* 2, the worst-case data age is only two frames.

For pedestrian tracking, *PMF* 3 was within at most 1.14% of *PMF* 2 in terms of *A-MOTA* across all scenarios. For vehicle tracking, this difference dropped to 0.73%. *PMF* 3 had *MOTP* within 4.95% that of *PMF* 2 across all scenarios and both target types. These results suggest that although it is better to have a lower worst-case age, the differences in both accuracy and precision are not extreme.

Q3: Is it better to have a higher chance of more recent data, or a lower worst-case age? *PMFs* 3 and 4 were chosen to answer this question: *PMF* 3 has more likely availability of the most recent results (probability 0.8 rather than 0.5), but has greater worst-case age (four frames versus three frames).

The results in columns *PMF* 3 and *PMF* 4 indicate that recency of available data is slightly more important than the worst-case data age. *PMF* 3 outperformed *PMF* 4 for *A-MOTA* and *MOTP* in seven out of eight comparisons each. However, the difference in these scores was only up to 0.84% for *A-MOTA* and 1.89% for *MOTP*. Therefore, although *PMF* 3 seems to perform slightly better, our experiments have not demonstrated a clear answer to this question.

Q4: How does the average case differ from the worst case? For the average and worst cases, we compare *PMFs* 1-4 to the corresponding worst-case *PMF* H_ps. We expect the average case to result in higher *A-MOTA* and *MOTP* scores, and for this difference to become more pronounced as the worst-case history age increases.

For a worst-case history age of two, we compare *PMF* 2 to *PMF* H_2: *PMF* 2 performed better than *PMF* H_2 in every comparison. Similarly, for worst-case history age of three frames, *PMF* 4 outperformed *PMF* H_3 in every comparison. As expected, for a worst-case history age of four frames, both *PMF* 1 and *PMF* 3 beat the worst-case *PMF* H_4 in every comparison, and for most by a large margin.

The history-versus-accuracy trade-off. The experimental results relating to question Q4 hint at our overall conclusion: allowing the infrequent use of older results in a MOT application has only minimal impact on the application's accuracy and precision, while allowing the computation of bounded response times for use in real-time certification. To explore this a little further, we make a final comparison against *PMF* H_2, which always uses the results of two frames prior, and thus corresponds to tracking using only half of the frames.

PMFs 1-4 are indexed in order of the expected results. That is, prior to performing experiments, we expected *PMF* 1 to perform the best and *PMF* 4 to perform the worst of this group. In fact, comparing these four PMFs to *PMF* H_2, we see that *PMF* H_2 did not perform as well as *PMF* 1 or *PMF* 2 in any comparison. Additionally, *PMF* H_2 scored better than *PMF* 3 or *PMF* 4 in only one of the 16 comparisons.

	Metric	*PMF* 1	*PMF* 2	*PMF* 3	*PMF* 4	*PMF* H_1	*PMF* H_2	*PMF* H_3	*PMF* H_4
Scenario 1	A-MOTA	**0.951**	**0.952**	**0.947**	0.939	**0.951**	0.927	0.893	0.888
	MOTP	0.709	0.700	0.669	0.670	**0.724**	0.644	0.588	0.567
Scenario 2	A-MOTA	**0.966**	0.962	0.955	0.951	**0.972**	0.933	0.943	0.933
	MOTP	**0.730**	0.724	0.699	0.697	**0.736**	0.669	0.609	0.560
Scenario 3	A-MOTA	**0.982**	**0.980**	**0.977**	0.974	**0.985**	0.973	0.954	0.931
	MOTP	0.721	0.714	0.689	0.686	**0.730**	0.656	0.628	0.580
Scenario 4	A-MOTA	**0.965**	**0.962**	0.957	0.952	**0.968**	0.939	0.934	0.908
	MOTP	**0.777**	0.772	0.751	0.751	**0.784**	0.741	0.690	0.650
Scenario 5	A-MOTA	**0.978**	**0.976**	0.970	0.968	**0.982**	0.957	0.942	0.904
	MOTP	0.652	0.646	0.614	0.607	**0.659**	0.575	0.511	0.483

TABLE III: Results for vehicles tracking using ground-truth detections. The best result in each row, as well as any within 1% of the best, are shown in bold.

	Metric	*PMF* 1	*PMF* 2	*PMF* 3	*PMF* 4	*PMF* H_1	*PMF* H_2	*PMF* H_3	*PMF* H_4
Scenario 1	A-MOTA	**0.884**	0.878	0.868	0.863	**0.892**	0.847	0.842	0.825
	MOTP	0.672	0.663	0.634	0.622	**0.683**	0.601	0.569	0.541
Scenario 2	A-MOTA	**0.968**	**0.969**	0.959	**0.962**	**0.971**	**0.962**	0.956	0.936
	MOTP	**0.808**	0.803	0.785	0.783	**0.814**	0.767	0.731	0.702
Scenario 4	A-MOTA	**0.936**	**0.934**	0.928	0.927	**0.938**	0.920	0.856	0.840
	MOTP	**0.794**	0.788	0.767	0.767	**0.800**	0.748	0.736	0.720

TABLE IV: Results for pedestrian tracking using ground-truth detections. (Note that Scenarios 3 and 5 did not include any pedestrians.) The best result in each row, as well as any within 1% of the best, are shown in bold.

Although results hold for both city- and highway-driving scenarios, the motion of the ego-vehicle does factor into the trade-off. Scenarios 1-4 feature city driving (of these, Scenario 3 involves the highest speed, as it does not involve the ego-vehicle turning sharply), and Scenario 5 occurs at highway speeds with only minor direction changes. As shown in Table III, direction changes were inversely correlated with *A-MOTA* scores: Scenarios 3 and 5 had much higher *A-MOTA* values for *PMFs* 1-4 and *PMF* H_1 than the other scenarios. For *MOTP*, the speed seemed to be the largest factor: Scenario 5 had much lower *MOTP* than all other scenarios.

C. Camera-Based Sensing

We have thus far examined the history-versus-accuracy trade-off in the presence of perfect detections, *i.e.* using the ground-truth bounding boxes of pedestrians and vehicles. However, in real-world scenarios, such ground-truth data are not available, which necessitates the use of CV-based object-detection algorithms.

We chose for a detector a state-of-the-art deep-learning model, Faster R-CNN [44], which has been shown to achieve a high level of accuracy. We used TensorFlow [1] to train a Faster R-CNN model with the Inception v2 feature extractor [26] (that was pre-trained on the COCO dataset [24], [34]) on a small dataset of 1300 images of bicycles, motorbikes, cars, and pedestrians generated from *CARLA* [10], [50].

We measured the detection accuracy of our model using a popular object-detection accuracy metric, the *mean average precision (mAP)* [19], [25]. After over $42,000$ training iterations, our Faster R-CNN model achieved a mAP score of 92.89%, indicating a high level of detection accuracy; a perfect object-detection algorithm would have a mAP score of 100%.

We now examine the impact of imperfect detections on tracking accuracy and precision. For this second set of accuracy experiments, we replaced the ground-truth vehicle and pedestrian detections with those generated by the Faster R-CNN model; the remainder of the experimental setup was unchanged. The resulting *A-MOTA* and *MOTP* values are given in Tables V and VI.

The impact of imperfect detections. We first compare the results using ground-truth data (Tables III and IV) with those from imperfect detections (Tables V and VI). As we would expect, using a detector decreased the accuracy of vehicle tracking and the precision of pedestrian tracking. Furthermore, *PMF* H_1 had the highest *MOTP* in each scenario for both vehicles and pedestrians. However, accuracy seems to be more affected by potentially incorrect detections. For example, *PMF* 4 resulted in the highest *A-MOTA* score for vehicles in Scenario 1, despite using older data with probability 0.5, and *PMF* H_4, which corresponds to always using data from four frames prior) had the highest *A-MOTA* score for pedestrians in Scenario 2. Altogether, our results suggest that tracking of pedestrians is less impacted by imperfect data, whereas for vehicles the impact can be quite large.

	Metric	*PMF* 1	*PMF* 2	*PMF* 3	*PMF* 4	*PMF* H_1	*PMF* H_2	*PMF* H_3	*PMF* H_4
Scenario 1	A-MOTA	0.741	0.785	0.810	**0.844**	0.761	0.727	0.698	0.659
	MOTP	**0.727**	**0.721**	0.664	0.696	**0.728**	0.642	0.584	0.561
Scenario 2	A-MOTA	0.833	**0.844**	0.810	0.815	**0.836**	**0.836**	**0.837**	0.833
	MOTP	**0.704**	0.689	0.673	0.649	**0.705**	0.631	0.593	0.547
Scenario 3	A-MOTA	**0.896**	0.874	0.883	0.877	**0.904**	0.858	0.850	0.836
	MOTP	0.601	0.586	0.556	0.573	**0.609**	0.532	0.506	0.496
Scenario 4	A-MOTA	**0.828**	0.818	0.793	0.802	**0.831**	0.817	0.810	0.789
	MOTP	**0.766**	0.757	0.711	0.712	**0.766**	0.712	0.688	0.656
Scenario 5	A-MOTA	**0.968**	**0.968**	**0.964**	**0.964**	**0.970**	0.960	**0.961**	0.949
	MOTP	0.747	0.728	0.703	0.703	**0.758**	0.659	0.579	0.536

TABLE V: Results for vehicle tracking using Faster R-CNN to detect vehicles. The best result in each row, as well as any within 1% of the best, are shown in bold.

	Metric	*PMF* 1	*PMF* 2	*PMF* 3	*PMF* 4	*PMF* H_1	*PMF* H_2	*PMF* H_3	*PMF* H_4
Scenario 1	A-MOTA	0.882	**0.888**	**0.894**	0.873	**0.890**	0.847	0.834	0.835
	MOTP	0.639	0.640	0.600	0.612	**0.654**	0.622	0.535	0.449
Scenario 2	A-MOTA	0.697	0.688	**0.717**	0.697	0.703	0.641	0.706	**0.723**
	MOTP	0.729	**0.745**	0.690	0.734	**0.751**	0.707	0.681	0.663
Scenario 4	A-MOTA	**0.938**	**0.938**	**0.934**	**0.938**	**0.938**	**0.938**	**0.936**	0.920
	MOTP	**0.763**	0.745	0.723	0.727	**0.769**	0.721	0.694	0.663

TABLE VI: Results for pedestrian tracking using Faster R-CNN to detect pedestrians. (Note that no pedestrians were present in Scenarios 3 and 5.) The best result in each row, as well as any within 1% of the best, are shown in bold.

Q1 through Q4, revisited. For question Q1, we again compare *PMF* 1 and *PMF* H_1. The *MOTP* score for *PMF* 1 was at most 2.93% lower than that of *PMF* H_1, indicating that there is still a precision loss due to having older data. Similarly for *A-MOTA*, the score for *PMF* 1 was at most 2.63% lower than that of *PMF* H_1.

In comparing *PMF* 2 and *PMF* 3, we see that having a higher worst-case history age (*PMF* 3) corresponded to much lower *MOTP* in all scenarios: *MOTP* was decreased by up to 7.91% compared to using *PMF* 2. However, *A-MOTA* was much more varied in the presence of a CV-based vehicle detector. In four of the eight comparisons, *PMF* 3 resulted in *higher* accuracy.

For the comparison for questions Q3, we again see different trends for precision relative to accuracy. In four of the eight comparisons, *PMF* 3 corresponded to lower accuracy than *PMF* 4; for *MOTP*, this number is six out of eight. Combined with the comparison for question Q2, this suggests that MOT precision relies on having lower worst-case history age, but MOT accuracy does not have such a clear dependence.

Finally, we revisit the comparisons related to question Q4. As with ground-truth detections, *PMFs* 1 and 3 usually outperformed *PMF* H_4, both by a large margin; for Scenario 2, however, this trend was surprisingly reversed. The differences for the other comparisons were less pronounced with CV-based detections than ground-truth detections: *PMF* 4 usually outperformed *PMF* H_3 in *A-MOTA* but had much higher *MOTP*, and the same trend held for *PMF* 2 and *PMF* H_2.

The trade-off, revisited. Compared to always having the most recent data (*i.e.*, *PMF* H_1), the lowest *A-MOTA* score among *PMFs* 1-4 was 4.57% reduced, and the lowest *MOTP* score was reduced by 8.79%. This suggests only a minor drop in accuracy, with a moderate drop in precision, as a result of using older data. It is also worth restating that the introduction of a real detector into the MOT pipeline makes these results less definitive, however: in some scenarios, using older data actually resulted in higher precision or accuracy.

However, if requiring $p = 1$ results in a system for which no response-time bounds can be computed, then measuring accuracy relative to *PMF* H_1 no longer has any relevance. Instead, we compare against *PMF* H_2, to compare to a system in which the history age of two is always used. In this case, both precision and accuracy are somewhat robust to a potentially imperfect detector. This is evident when comparing *PMF* H_2 to *PMFs* 1-2, which correspond to situations in which the most recent data are available with probability at least 0.8 and the the worst-case age is two with probability at least 0.99. Aside from a single comparison in which *PMF* H_2 has slightly higher accuracy, when using our Faster R-CNN model to detect vehicles and pedestrians, *PMFs* 1 and 2 had 0.12-8.74% higher accuracy than that of *PMF* H_2 and 2.73-13.35% higher tracking precision. These results suggest that for a system that would otherwise be unschedulable, the increased worst-case history requirement due to increased parallelism only slightly reduces MOT accuracy and precision, and that reduction can be greatly mitigated if the most recent results

are usually available.

VII. Case Study

The history-versus-response-time experiments in Sec. III indicate that increasing P_i past the minimum required to ensure that $P_i > u_i$ can significantly reduce response-time bounds. However, the experiments in Sec. VI showed that the impact of increasing P_i depends not only on the absolute value of P_i, but also the distribution of available historical results.

What remains to be seen is what history distributions can occur in practice, and how this impacts the accuracy of the MOT application. We now present the results of a case study experiment including several automotive applications running alongside an MOT tracking application, all running as real-time tasks under global EDF scheduling using LITMUS$^{\mathrm{RT}}$ [5], [8], a research-based modification of the Linux kernel to enable real-time scheduling.

A. Experimental Setup

We scheduled three additional applications running alongside the deep-learning-based MOT application discussed in Sec. VI-C. The first two perform global and local path planning via Rapidly-exploring Random Trees (RRT) [30] and the Artificial Potential Field (APF) method [29], and the third implements lane detection using the Hough Transform.

For this case study, we consider Scenario 2, in which both vehicles and pedestrians are tracked by a camera attached to a vehicle moving at city-driving speeds. Thus, we assume the vehicle travels at around 30 miles per hour, or 44 feet per second. For the global-planner task, we selected a period of 5 seconds (corresponding to re-planning the global path every 220 feet). The local planner determines the more immediate path, so we selected for it a period of 200 milliseconds, corresponding to approximately every 9 feet. We chose a period of 200 milliseconds for lane detection as well. The deep-learning-based detector used in our MOT application takes around 2 seconds to process a single image. Therefore, we chose a period of 2 seconds for both the pedestrian- and vehicle-MOT tasks. For all tasks, we assume implicit deadlines, *i.e.*, that the deadline equals the period.

We performed our case study on a platform with a four-core 3.50 GHz Intel i5-6600K processor and one NVIDIA GeForce GTX 980 Ti GPU. Each CPU core has a 32-KB L1 instruction cache, a 32-KB L1 data cache, and a 256-KB L2 cache; all four CPU cores share a 6-MB L3 cache.

B. Distribution of Available History

To determine the distribution of available history for the MOT tasks, we set $P_i = 4$ and performed tracking of both vehicles and pedestrians for the 300 frames of Scenario 2. The most recent result was available for 195 of the 300 frames, the second-most recent for 75 frames, and third-most and fourth-most recent were available for 18 and 12 frames, respectively. The resulting distribution, represented as a tuple, is $(0.65, 0.25, 0.06, 0.04)$.

C. History-versus-Accuracy

Given the results discussed in Sec. VI, we assume the accuracy and precision for this observed history distribution should be higher than that of *PMF* 4, as the most-recent data is available more frequently than half the time. However, we expect that the observed distribution should not perform as well as the other PMFs. After running Scenario 2 100 times and sampling from our observed history distribution, we observed an *A-MOTA* score of 0.842 and a *MOTP* score of 0.654 for vehicle tracking. As expected, both accuracy and precision were higher than that of *PMF* 4. In fact, the accuracy for our observed history distribution was within 1% of the best (0.844 for *PMF* 2).

Given the results of this case study, we believe that the results presented in Sec. VI can be used as general guidance for real task systems: as long as the most recent historical results are available most of the time, the accuracy drop is minimal compared to sequential scheduling.

VIII. Conclusion

Prior work by Amert *et al.* [2] has shown that response-time bounds can be computed for cyclic real-time processing graphs if $p > 1$ is allowed for each cycle with utilization exceeding 1.0. However, $p > 1$ permits non-immediate back history to be used, which in the context of CV tracking applications could potentially compromise tracking accuracy. In this paper, we have studied this issue in detail. We first showed that increasing p greatly reduced response-time bounds for synthetically generated graph-based task systems. In the context of our accuracy study, we found that allowing $p > 1$ did not significantly affect accuracy, as long as immediate back history is frequently available. This is a favorable observation from a real-time schedulability point of view because it points to the existence of a certain amount of leeway in graph scheduling that can be reasonably exploited. It is worth noting that, while approaches that lessen CV accuracy should be generally eschewed, *accuracy loss is mainly being contemplated here to ensure the schedulability of a graph that would otherwise be unschedulable.*

In our accuracy study, accuracy was assessed using well-established metrics pertaining to CV algorithms. In future work, we intend to fully integrate the usage of relaxed back-history requirements within the control and decision-making components of CARLA, and to re-assess the impact of relaxing such requirements in actual driving scenarios via higher-level metrics such as the number of accidents or traffic violations. CARLA is a complex system, so this integration will be a major undertaking.

References

[1] M. Abadi, P. Barham, J. Chen, Z. Chen, A. Davis, J. Dean, M. Devin, S. Ghemawat, G. Irving, M. Isard, M. Kudlur, J. Levenberg, R. Monga, S. Moore, D. G. Murray, B. Steiner, P. Tucker, V. Vasudevan, P. Warden, M. Wicke, Y. Yu, and X. Zheng, "TensorFlow: A system for large-scale machine learning," in *Proceedings of the 12th USENIX Symposium on Operating Systems Design and Implementation*, 2016, pp. 265–283.

[2] T. Amert, S. Voronov, and J. H. Anderson, "OpenVX and real-time certification: The troublesome history," in *Proceedings of the 40th IEEE Real-Time Systems Symposium*, 2019, pp. 312–325.

[3] P. Bergmann, T. Meinhardt, and L. Leal-Taixe, "Tracking without bells and whistles," *arXiv preprint arXiv:1903.05625*, 2019.

[4] M. Bleyer, C. Rhemann, and C. Rother, "Patchmatch stereo-stereo matching with slanted support windows," in *Proceedings of the British Machine Vision Conference*, vol. 11, 2011, pp. 1–11.

[5] B. B. Brandenburg, "Scheduling and locking in multiprocessor real-time operating systems," Ph.D. dissertation, University of North Carolina at Chapel Hill, Chapel Hill, NC, USA, 2011.

[6] M. D. Breitenstein, F. Reichlin, B. Leibe, E. Koller-Meier, and L. Van Gool, "Robust tracking-by-detection using a detector confidence particle filter," in *Proceedings of the 12th IEEE International Conference on Computer Vision*, 2009, pp. 1515–1522.

[7] C. Cadena, L. Carlone, H. Carrillo, Y. Latif, D. Scaramuzza, J. Neira, I. Reid, and J. J. Leonard, "Past, present, and future of simultaneous localization and mapping: Toward the robust-perception age," *IEEE Transactions on Robotics*, vol. 32, no. 6, pp. 1309–1332, 2016.

[8] J. Calandrino, H. Leontyev, A. Block, U. C. Devi, and J. H. Anderson, "LITMUS^{RT} : A testbed for empirically comparing real-time multiprocessor schedulers," in *Proceedings of the 27th IEEE International Real-Time Systems Symposium*, 2006, pp. 111–126.

[9] N. Dalal and B. Triggs, "Histograms of oriented gradients for human detection," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2005, pp. 886–893 Vol. 1.

[10] DanielHfnr, "Carla object detection dataset," https://github.com/DanielHfnr/Carla-Object-Detection-Dataset, 2019 (accessed 10 September 2020).

[11] U. Devi and J. H. Anderson, "Tardiness bounds under global EDF scheduling on a multiprocessor," *Real-Time Systems*, vol. 38, no. 2, pp. 133–189, 2008.

[12] A. Dosovitskiy, G. Ros, F. Codevilla, A. Lopez, and V. Koltun, "CARLA: An open urban driving simulator," in *Proceedings of the 1st Annual Conference on Robot Learning*, 2017, pp. 1–16.

[13] D. Eigen, C. Puhrsch, and R. Fergus, "Depth map prediction from a single image using a multi-scale deep network," in *Proceedings of Advances in Neural Information Processing Systems*, 2014, pp. 2366–2374.

[14] J. Engel, T. Schöps, and D. Cremers, "LSD-SLAM: Large-scale direct monocular SLAM," in *Proceedings of the European Conference on Computer Vision*, 2014, pp. 834–849.

[15] Epic Games, "Unreal Engine," https://www.unrealengine.com, 2020 (accessed 10 September 2020).

[16] P. Erdős and A. Rényi, "On random graphs i," *Publicationes Mathematicae*, vol. 6, pp. 290–297, 1958.

[17] J. P. Erickson and J. H. Anderson, "Response time bounds for G-EDF without intra-task precedence constraints," in *Proceedings of the 15th International Conference On Principles Of Distributed Systems*, 2011, pp. 128–142.

[18] J. P. Erickson, N. Guan, and S. Baruah, "Tardiness bounds for global EDF with deadlines different from periods," in *Proceedings of the 14th International Conference On Principles Of Distributed Systems*, 2010, pp. 286–301.

[19] M. Everingham, L. Van Gool, C. K. Williams, J. Winn, and A. Zisserman, "The Pascal visual object classes (VOC) challenge," *International Journal of Computer Vision*, vol. 88, no. 2, pp. 303–338, 2010.

[20] J.-M. Frahm, P. Fite-Georgel, D. Gallup, T. Johnson, R. Raguram, C. Wu, Y.-H. Jen, E. Dunn, B. Clipp, S. Lazebnik, and M. Pollefeys, "Building Rome on a cloudless day," in *Proceedings of the European Conference on Computer Vision*, 2010, pp. 368–381.

[21] R. Girshick, J. Donahue, T. Darrell, and J. Malik, "Rich feature hierarchies for accurate object detection and semantic segmentation," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2014, pp. 580–587.

[22] J. Heinly, J. L. Schönberger, E. Dunn, and J.-M. Frahm, "Reconstructing the World* in Six Days *(As Captured by the Yahoo 100 Million Image Dataset)," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2015, pp. 3287–3295.

[23] H. Hirschmuller, "Stereo processing by semiglobal matching and mutual information," *IEEE Transactions on pattern analysis and machine intelligence*, vol. 30, no. 2, pp. 328–341, 2008.

[24] J. Huang, V. Rathod, C. Sun, M. Zhu, A. Korattikara, A. Fathi, I. Fischer, Z. Wojna, Y. Song, S. Guadarrama *et al.*, "Speed/accuracy trade-offs for modern convolutional object detectors," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2017, pp. 7310–7311.

[25] J. Hui, "mAP (mean Average Precision) for object detection," https://medium.com/@jonathan_hui/map-mean-average-precision-for-object-detection-45c121a31173, 2018 (accessed 10 September 2020).

[26] S. Ioffe and C. Szegedy, "Batch normalization: Accelerating deep network training by reducing internal covariate shift," *arXiv preprint arXiv:1502.03167*, 2015.

[27] M. Isard and A. Blake, "Condensation—conditional density propagation for visual tracking," *International journal of computer vision*, vol. 29, no. 1, pp. 5–28, 1998.

[28] H. Kataoka, K. Tamura, K. Iwata, Y. Satoh, Y. Matsui, and Y. Aoki, "Extended feature descriptor and vehicle motion model with tracking-by-detection for pedestrian active safety," *IEICE TRANSACTIONS on Information and Systems*, vol. 97, no. 2, pp. 296–304, 2014.

[29] O. Khatib, "Real-time obstacle avoidance for manipulators and mobile robots," in *Autonomous robot vehicles*. Springer, 1986, pp. 396–404.

[30] S. M. LaValle and J. J. Kuffner Jr, "Randomized kinodynamic planning," *The international journal of robotics research*, vol. 20, no. 5, pp. 378–400, 2001.

[31] H. Leontyev and J. H. Anderson, "Generalized tardiness bounds for global multiprocessor scheduling," in *Proceedings of the 28th IEEE Real-Time Systems Symposium*, 2007, pp. 413–422.

[32] ——, "Tardiness bounds for FIFO scheduling on multiprocessors," in *Proceedings of the 19th Euromicro Conference on Real-Time Systems*, 2007, pp. 71–80.

[33] Z. Li and N. Snavely, "MegaDepth: Learning single-view depth prediction from internet photos," *arXiv preprint arXiv:1804.00607*, 2018.

[34] T. Lin, M. Maire, S. J. Belongie, L. D. Bourdev, R. B. Girshick, J. Hays, P. Perona, D. Ramanan, P. Dollár, and C. L. Zitnick, "Microsoft COCO: Common objects in context," *CoRR*, vol. abs/1405.0312, 2014. [Online]. Available: http://arxiv.org/abs/1405.0312

[35] C. Liu and J. H. Anderson, "Supporting graph-based real-time applications in distributed systems," in *Proceedings of the 20th IEEE International Conference on Embedded and Real-Time Computing Systems and Applications*, 2011, pp. 143–152.

[36] A. Milan, L. Leal-Taixé, I. Reid, S. Roth, and K. Schindler, "MOT16: A benchmark for multi-object tracking," *arXiv preprint arXiv:1603.00831*, 2016.

[37] R. Mur-Artal, J. M. M. Montiel, and J. D. Tardos, "ORB-SLAM: A versatile and accurate monocular slam system," *IEEE Transactions on Robotics*, vol. 31, no. 5, pp. 1147–1163, 2015.

[38] R. Mur-Artal and J. D. Tardós, "ORB-SLAM2: An open-source slam system for monocular, stereo, and RGB-D cameras," *IEEE Transactions on Robotics*, vol. 33, no. 5, pp. 1255–1262, 2017.

[39] S. Murray, "Real-time multiple object tracking-a study on the importance of speed," *arXiv preprint arXiv:1709.03572*, 2017.

[40] W. G. Najm, J. D. Smith, and M. Yanagisawa, "Pre-crash scenario typology for crash avoidance research," United States. National Highway Traffic Safety Administration, Tech. Rep., 2007.

[41] K. Okuma, A. Taleghani, N. De Freitas, J. J. Little, and D. G. Lowe, "A boosted particle filter: Multitarget detection and tracking," in *European conference on computer vision*. Springer, 2004, pp. 28–39.

[42] F. Porikli and O. Tuzel, "Object tracking in low-frame-rate video," in *Image and Video Communications and Processing 2005*, vol. 5685. International Society for Optics and Photonics, 2005, pp. 72–79.

[43] T. Price, J. L. Schönberger, Z. Wei, M. Pollefeys, and J.-M. Frahm, "Augmenting crowd-sourced 3D reconstructions using semantic detections," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2018, pp. 1926–1935.

[44] S. Ren, K. He, R. Girshick, and J. Sun, "Faster R-CNN: Towards real-time object detection with region proposal networks," in *Advances in neural information processing systems*, 2015, pp. 91–99.

[45] H. Rezatofighi, N. Tsoi, J. Gwak, A. Sadeghian, I. Reid, and S. Savarese, "Generalized intersection over union: A metric and a loss for bounding box regression," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2019, pp. 658–666.

[46] D. Scharstein, H. Hirschmüller, Y. Kitajima, G. Krathwohl, N. Nešić, X. Wang, and P. Westling, "High-resolution stereo datasets with subpixel-accurate ground truth," in *Proceedings of the German Conference on Pattern Recognition*, 2014, pp. 31–42.

[47] J. L. Schönberger and J.-M. Frahm, "Structure-from-motion revisited," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2016, pp. 4104–4113.
[48] R. Schubert, E. Richter, and G. Wanielik, "Comparison and evaluation of advanced motion models for vehicle tracking," in *Proceedings of the 11th IEEE International Conference on Information Fusion*, 2008, pp. 1–6.
[49] R. Stiefelhagen, K. Bernardin, R. Bowers, J. Garofolo, D. Mostefa, and P. Soundararajan, "The CLEAR 2006 evaluation," in *Proceedings of the 1st International Evaluation Workshop on Classification of Events, Activities and Relationships*, 2006, pp. 1–44.
[50] tkortz, "Carla object detection dataset," https://github.com/tkortz/Carla-Object-Detection-Dataset, 2020 (accessed 10 September 2020).
[51] B. Ummenhofer, H. Zhou, J. Uhrig, N. Mayer, E. Ilg, A. Dosovitskiy, and T. Brox, "DeMoN: Depth and motion network for learning monocular stereo," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2017, pp. 5038–5047.
[52] K. Yang, G. A. Elliott, and J. H. Anderson, "Analysis for supporting real-time computer vision workloads using OpenVX on multicore+GPU platforms," in *Proceedings of the 23rd International Conference on Real-Time Networks and Systems*, 2015, pp. 77–86.
[53] M. Yang, T. Amert, K. Yang, N. Otterness, J. H. Anderson, F. D. Smith, and S. Wang, "Making OpenVX really 'Real Time'," in *Proceedings of the 39th IEEE Real-Time Systems Symposium*, 2018, pp. 80–93.
[54] J. Zbontar and Y. LeCun, "Stereo matching by training a convolutional neural network to compare image patches," *Journal of Machine Learning Research*, vol. 17, no. 1-32, p. 2, 2016.
[55] T. Zhou, M. Brown, N. Snavely, and D. G. Lowe, "Unsupervised learning of depth and ego-motion from video," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2017, pp. 6612–6619.