The Price of Schedulability in Multi-Object Tracking: The History-vs.-Accuracy Trade-Off*

Tanya Amert, Ming Yang, 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.-accuracy trade-off that it exposes. In this paper, this trade-off is explored in depth through 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 in this study.**

***Index Terms*—autonomous driving, 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. 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 $p = 1$.

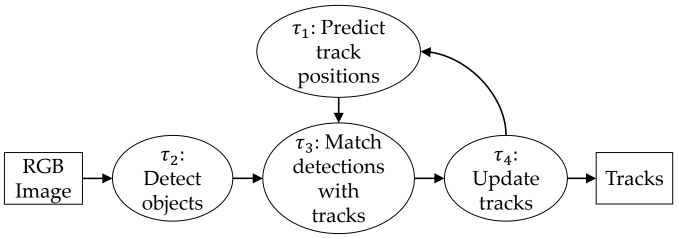

RGB Image
τ_2: Detect objects
τ_1: Predict track positions
τ_3: Match detections with tracks
τ_4: Update tracks
Tracks

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

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 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

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

[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."

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 [34], 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 [36], [37] 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, CV accuracy may 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.

Study overview. We consider an MOT system in which pedestrians and other vehicles are tracked via images recorded by a camera attached to a moving vehicle. In order to observe the impact of increasing p on the performance of 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 [10], an open-source simulator designed for research on autonomous-driving systems. The use of CARLA allows 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.

Organization. The remainder of this paper is organized as follows. In Sec. II, we discuss the real-time scheduling implications of varying parallelism, and then describe the MOT pipeline in detail. Next, we give an overview of our evaluation of the history-versus-accuracy trade-off in Sec. III. In Sec. IV we discuss the results of our evaluation, both when ground-truth detections are used and in the presence of a CV-based object detector, and conclude in Sec. V.

II. Background

In this section, we provide necessary background on the real-time scheduling of graphs containing cycles and on a Tracking-By-Detection MOT pipeline

A. Real-Time Graph Scheduling

Much prior work on response-time analysis under global schedulers [9], [15], [27], [28] has assumed the sequential sporadic task model, which requires invocations of the same task to execute sequentially, *i.e.*, no two invocations of task τ_1 in Fig. 1 may execute concurrently. This model is in contrast to the fully parallel task model, in which any number of invocations of a given task may execute concurrently. Full intra-task parallelism has been shown to enable much smaller response-time bounds to be guaranteed [14].

Ex. 1. Most prior work on response-time analysis for graph-based workloads applies only for DAGs. Therefore, the cycle in Fig. 1 must be replaced with a single "supernode," which we call τ_c. Assume, for this example, that the supernode has $p = 2$. Fig. 2 depicts possible schedules for invocations (called *jobs*) of this supernode on a platform with four CPUs. Each job $J_{c,k}$ is released at time $r_{c,k}$.

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

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

Unfortunately, unrestricted intra-task parallelism can violate the dependencies required by back edges. However, sequential execution can result in a system that is unschedulable.

Ex. 1 (cont'd). The supernode τ_c was created from a cycle with $p = 2$. Thus, job $J_{c,23}$ requires output from job $J_{c,21}$. However, in schedule (b) of Fig. 2, jobs $J_{c,21}$ and $J_{c,23}$ execute concurrently, violating this precedence constraint. If jobs execute sequentially as in Fig. 2(a), response times can be unbounded for τ_c if the cycle's utilization exceeds 1.0. $\Diamond$

[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.

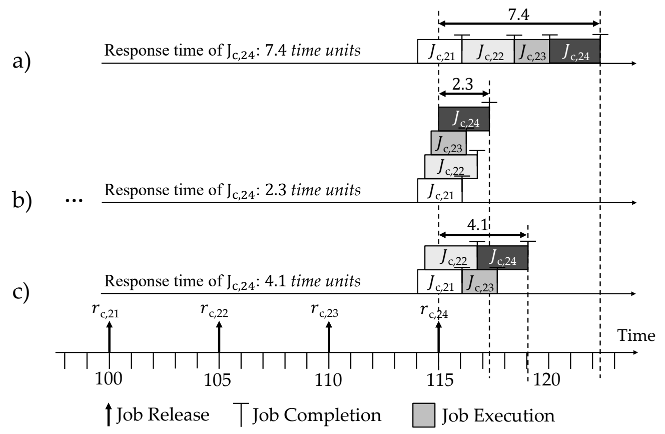


Fig. 2: Scheduling repercussions of the degree of intra-task parallelism, including a) sequential execution, b) fully parallel execution, and c) restricted parallelism. Successive jobs are shaded progressively darker. Assume the depicted jobs are scheduled alongside other jobs, which are not shown.

Under the sporadic task model, a task τ_i is 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. The *restricted parallelism* 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. Additionally, the model requires that the utilization of any cycle is at most P_i. 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.

In this paper, we assume $P_i = p$, although any $P_i \leq p$ guarantees that prior-history requirements are met.

Ex. 1 (cont'd). Restricted intra-task parallelism ($P_c = 2$) is shown in schedule (c) of Fig. 2. The response time of $J_{c,24}$ is increased to 4.1, but the history requirements are respected, as only $p = 2$ jobs of τ_c execute concurrently. ◊

B. 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 in detail, and discuss a few representative approaches for the implementation of each step.

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. 2. We demonstrate the steps of the TBD pipeline via a continuing example. Fig. 3 depicts the results of each of the four TBD steps (the order matches the task indices in Fig. 1) for a given frame. The input RGB image contains three vehicles, one of which is occluded behind another (center right). There are four tracks (indicated by dots showing position estimates over time), one for each visible vehicle, plus a fourth for a vehicle that is out of view to the right.

Dashed boxes in Fig. 3a 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 [43], particle filtering [5], [25], or optical flow [26].

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*.

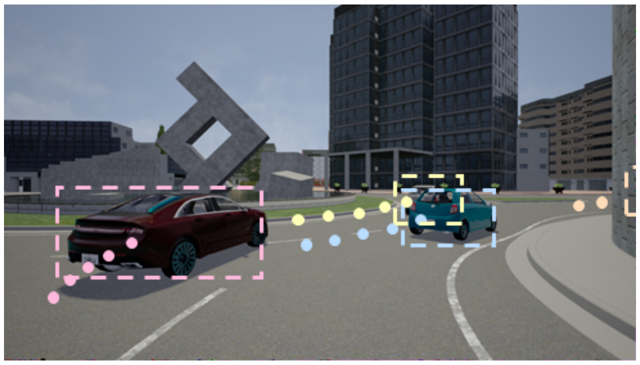
Ex. 2 (cont'd). The detection step outputs a bounding box for each detected vehicle, as in Fig. 3b. In this example, one vehicle is occluded (center right) and the other is off screen (far right), so only two bounding boxes are outputted. ◊

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 feature selection method is histogram of oriented gradients (HOG) [7]. Recently, deep convolutional neural networks have been used [18], [22]. Popular classifiers include linear support vector machines (SVMs), as originally used with HOG [7], or latent SVMs [19].

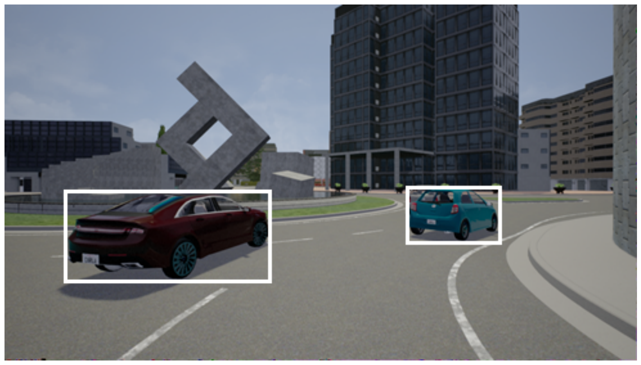
Matching detections to tracks. Given a set of detected bounding boxes and predictions of new track positions, the percentage overlap is compared for all detection-prediction rectangle pairs. The Hungarian method (also known as Munkres' algorithm) can be used to quickly match detections to predictions [31], [44]. The overlap of two rectangles is computed using the *intersection-over-union* measure (IOU) [40], 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. 2 (cont'd). The bounding boxes computed by the two prior steps are used to calculate pair-wise overlap ratios. As shown in Fig. 3c, the detections and predictions for the two unoccluded vehicles are closely aligned; the cars on the left and right have IoUs of 0.85 and 0.81, respectively, indicating strong matches. Two predictions are unmatched, one corresponding to the occluded vehicle and the other to the vehicle that exited the scene, and no detections are unmatched. ◊

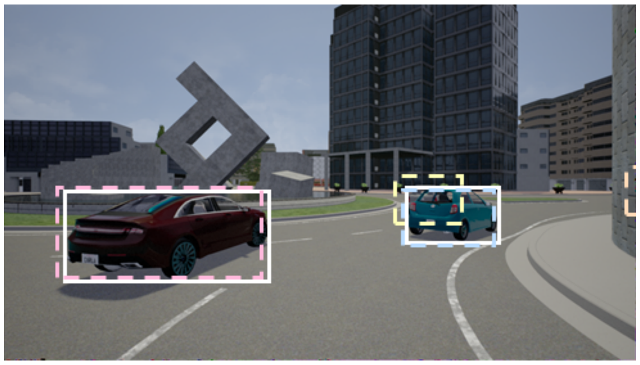
Updating the tracks. For each prediction that is matched with a detection, the corresponding track is updated based on the detected position. Depending on the motion model, the model is also updated based on the new position. If a track



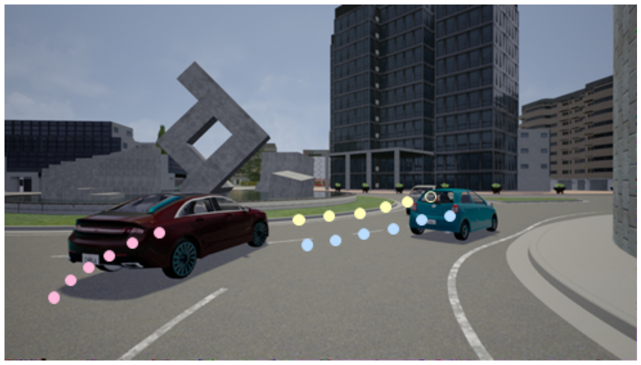
(a) Each track (the output of processing the previous frame) is extrapolated to the current frame based on the choice of motion model. In this example, four tracks (indicated by dots) are used to predict the positions of four vehicles in the current frame; three vehicles are visible, and one is out of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



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

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. 2 (cont'd). Tracks corresponding to the two matched predictions are updated to contain a new position based on the detection, as shown in Fig. 3d. The occluded vehicle's track prediction is not matched, so its track is updated based on the prediction, indicated in Fig. 3d by the empty circle. The predictions of the track on the far right have not been matched for many frames, so that track is deleted. ◊

C. MOT from a Moving Vehicle

The pipeline described in Sec. II-B 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 [6], [12], [32], [33] and *structure-from-motion (SfM)* methods [17], [20], [38], [42]. 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], [11], [21], [29], [41], [45]–[47] 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.

D. 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 results of our evaluation in Sec. IV, we consider first a system with ground-truth detections in order to evaluate tracking is isolation, and then a system with CV-based detections. In the next section, we describe our experimental evaluation in full.

III. Evaluating the History-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* [10], an open-source urban-driving simulator. *CARLA* uses Unreal Engine 4 [13] 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 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 [35], 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.

The four scenarios are depicted in Fig. 4. 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.

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.

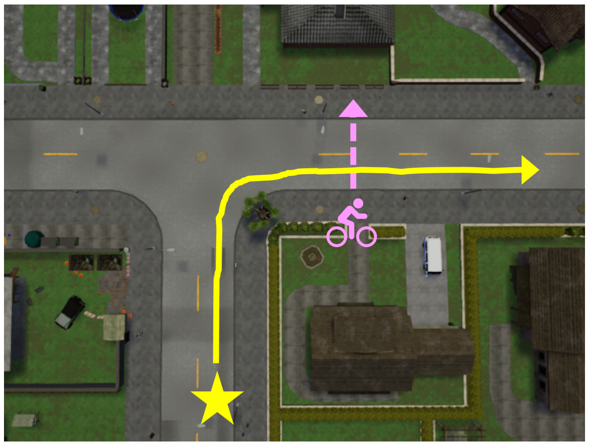
We evaluated eight PMFs, listed in Table I, chosen to answer four questions:

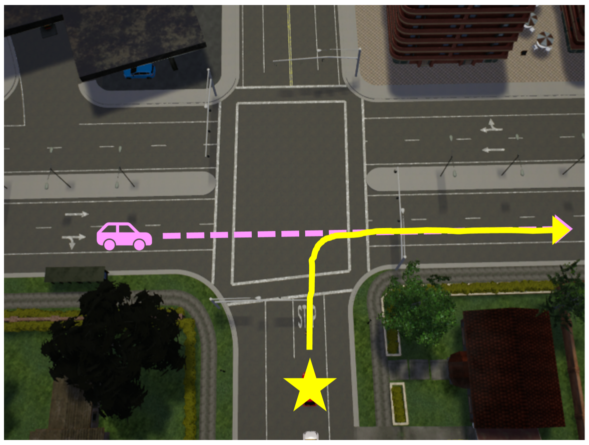
Q1 What if the most recent data is sometimes unavailable?
Q2 How much of an impact does the worst-case age have if the most recent data is usually available?
Q3 Is it better to have a higher chance of more recent data, or a lower worst-case age?
Q4 How does the average case differ from the worst case?

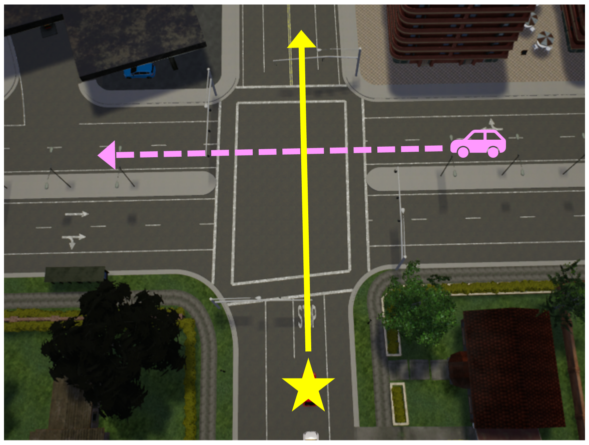
Comparing *PMFs* 1 and H_1 should answer question Q1. We can answer question Q2 by comparing *PMFs* 2 and 3. For question Q3 we compare *PMFs* 3 and 4. Finally, we compare *PMFs* 1-4 to the corresponding worst-case *PMF* H_p (*e.g.*, *PMFs* 4 and H_3) to answer question Q4.

IV. Results

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


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


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


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

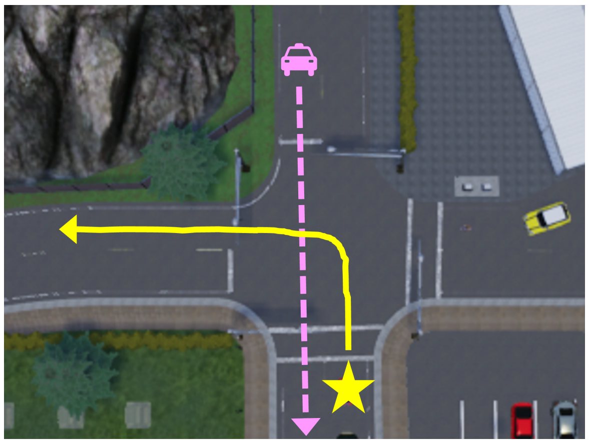

(d) Unprotected left turn at an intersection with oncoming traffic.

Fig. 4: The four 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.

if from a perfect detector), and the 3D distance to each target within the scene (corresponding to perfect stereo estimation). (In Sec. IV-C, we remove the assumption of a perfect detector.)

A. *Evaluation Metrics for MOT*

In this section, we provide an overview of the standard metrics used to evaluate MOT applications [44]. We consider first the metrics defined for each frame, then the metrics for each trajectory, and finally 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.

	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	$(0, \overset{p-1}{\dots}, 0, 1)$

TABLE I: 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, $0, \overset{p-1}{\dots}, 0$ 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.

The final per-frame metric is the number of ID switches. We use the stricter definition given by Milan *et al.* [31]: an *ID switch* (IDSW_t) occurs for frame t when a target is assigned a track that is different from its prior track assignment. In the best case, $\mathit{TP}_t = \mathit{GT}_t$ and $\mathit{FN}_t = \mathit{FP}_t = \mathit{IDSW}_t = 0$.

Per-trajectory metrics. In addition to per-frame metrics, we evaluate the results of tracking for each ground-truth trajectory. Following [44], we classify each target as *mostly tracked (MT)* if it is successfully tracked (*i.e.*, its detections are matched to track predictions) for at least 80% of the frames for which it is present, *mostly lost (ML)* if it is tracked successfully for at most 20% of the frames for which it is present, and *partially tracked (PT)* otherwise. We can also measure the continuity of each track by considering the track's *fragmentation count* (*FM*). A fragmentation occurs when a target's trajectory becomes untracked and later becomes tracked again.

With an ideal tracker, all targets should be mostly tracked, and thus we would expect to find $PT = ML = 0$. Additionally, we would also expect $FM = 0$.

Overall metrics. A number of overall metrics are standardly used in the CV literature to evaluate a tracker's performance holistically. The *Multiple Object Tracking Accuracy (MOTA)* metric combines information about the false positives, false negatives, and ID switches:

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

For an ideal tracker, $MOTA = 1.0$. However, if maintaining the correct ID of each tracked object is not needed for a given application, the *MOTA* might not appropriately represent the success of the tracker. Instead, *A-MOTA* can be used in this case, as it ignores ID switches:

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

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:

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

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

Q1: What if the most recent data is 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 is always available.

Comparing the two columns in Tables II and III, 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. The average *FM* count is only slightly larger for *PMF* 1 in all scenarios as well.

Q2: How much of an impact does the worst-case age have if the most recent data is usually available? This question can be answered by comparing *PMFs* 2 and 3. For both distributions, the most recent data is 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%. *MOTP* was more consistent between the two; *PMF* 3 had *MOTP* within 4.43% that of *PMF* 2 across all scenarios and both target types. Additionally, *PMF* 3 resulted in up to 2.5 times as many track fragmentations than *PMF* 2. 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. However, higher worst-case age does contribute to more track fragmentations.

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 six out of seven comparisons each. However, the difference in these scores was only up to 0.84% for *A-MOTA* and 1.89% for *MOTP*. On the other hand, *PMF* 3 resulted in higher average *FM* count in five out of seven comparisons, although in the worst case *PMF* 3 only had 10.18% more track fragmentations. 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

	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
	Avg. FM	1.117	1.160	1.503	1.350	**1.000**	1.667	2.000	2.667
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
	Avg. FM	0.590	0.688	1.077	1.070	**0.500**	1.500	1.833	1.667
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
	Avg. FM	0.553	0.642	1.110	1.058	**0.500**	1.333	1.833	2.000
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
	Avg. FM	1.583	1.622	1.985	1.948	**1.500**	2.500	2.500	2.333

TABLE II: 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
	Avg. FM	0.853	0.888	1.324	1.237	**0.667**	1.917	1.583	1.667
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
	Avg. FM	0.225	0.220	0.568	0.590	**0.000**	1.000	1.000	1.000
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
	Avg. FM	0.135	0.178	0.445	0.525	**0.000**	1.000	1.000	1.000

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

corresponding worst-case *PMF* H_ps. We expect the average case to result in higher *A-MOTA* and *MOTP* scores and lower average *FM* counts, 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 twenty-one comparisons.

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 is 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 [39], 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 [24] (that was pre-trained on the COCO dataset [22], [30]) on a small dataset of 2000 images of bicycles, motorbikes, and cars generated from *CARLA* [8].

We measured the detection accuracy of our model using a popular object detection accuracy metric, the *mean average precision (mAP)* [16], [23]. After $40,000$ training iterations, our Faster R-CNN model achieved a mAP score of 93.14%,

	Metric	*PMF* 1	*PMF* 2	*PMF* 3	*PMF* 4	*PMF* H_1	*PMF* H_2	*PMF* H_3	*PMF* H_4
Scenario 3	A-MOTA	**0.982**	**0.980**	**0.977**	0.974	**0.985**	0.973	0.954	0.931
(ground truth)	MOTP	0.721	0.714	0.689	0.686	**0.730**	0.656	0.628	0.580
Scenario 3	A-MOTA	0.899	0.880	**0.911**	0.879	**0.905**	0.875	0.878	0.865
(R-CNN detections)	MOTP	0.593	0.590	0.588	0.567	**0.614**	0.554	0.521	0.497

TABLE IV: Results for vehicle tracking using Faster R-CNN to detect vehicles, compared to the results using ground-truth detections (copied from Table II). The best result in each row, as well as any within 1% of the best, are shown in bold.

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. We replaced the ground-truth vehicle detections with those generated by the Faster R-CNN model; the remainder of the experimental setup was unchanged. For this comparison, we evaluated Scenario 3, as there were no pedestrians present in that scenario. The resulting *A-MOTA* and *MOTP* values are given in Table IV; without a separate mapping step of ground-truth trajectories to the detections provided by the Faster R-CNN model, we did not compute *FM* counts a for this scenario.

Q1 through Q4, revisited. For question Q1, we again compare *PMF* 1 and *PMF* H_1. The MOTP score for *PMF* 1 was 3.42% lower than that of *PMF* H_1, indicating that there is still a precision loss due to having older data. However, accuracy seems to be more affected by potentially incorrect detections: somewhat unexpectedly, *PMF* 3 resulted in a better *A-MOTA* score than both *PMF* 1 and *PMF* H_1.

In comparing *PMF* 2 and *PMF* 3, we see that the worst-case age has less impact on *MOTP* than *A-MOTA* in the presence of a CV-based vehicle detector. Despite having a higher worst-case history age, *PMF* 3 resulted in higher accuracy. However, *PMF* 2 had a slightly higher precision than *PMF* 3.

When considering the comparison for question Q3, we see that in the presence of CV-based object detections, it is clearly better to have the most recent data available more frequently than to have a lower worst-case age: *PMF* 4 had 3.51% lower *A-MOTA* and 3.57% lower *MOTP* than *PMF* 3.

Finally, we revisit the comparisons related to question Q4. As with ground-truth detections, *PMFs* 1 and 3 outperformed *PMF* H_4, both by a large margin. The differences for the other comparisons were less pronounced with CV-based detections than ground-truth detections: *PMF* 4 barely 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 was 2.87% reduced, and the lowest *MOTP* score was reduced by 7.65%. This suggests only a small drop in accuracy, with a moderate drop in precision, as a result of using older data.

However, if requiring $p = 1$ results in a system for which no response-time bounds can be computed, then measuring accuracy no longer has any relevance. Instead, we compare against *PMF* H_2, to compare to a system in which the worst-case 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* 1-4. When using our Faster R-CNN model to detect vehicles, *PMFs* 1-4 had 0.46-4.11% higher accuracy than that of *PMF* H_2, and the tracking precision for *PMFs* 1-4 was 2.35-7.04% higher than that of *PMF* H_2.

V. 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 any 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. In the context of our 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 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. CARLA is a complex system, so this integration will be a major undertaking. Additionally, our study considered only city driving scenarios. We plan to extend our assessment to include highway driving scenarios in order to explore the impact of the speed of the ego-vehicle on the history-versus-accuracy trade-off.

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 *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 British Machine Vision Conference*, vol. 11, 2011, pp. 1–11.

[5] 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 *2009 IEEE 12th International Conference on Computer Vision*. IEEE, 2009, pp. 1515–1522.

[6] 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.

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

[8] DanielHfnr, “Carla object detection dataset,” Online at https://github.com/DanielHfnr/Carla-Object-Detection-Dataset, 2019.

[9] 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.

[10] 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.

[11] 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.

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

[13] Epic Games, “Unreal Engine,” Online at https://www.unrealengine.com.

[14] 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.

[15] 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.

[16] 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.

[17] 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 European Conference on Computer Vision*, 2010, pp. 368–381.

[18] 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.

[19] R. Girshick, F. Iandola, T. Darrell, and J. Malik, “Deformable part models are convolutional neural networks,” in *Proceedings of the IEEE conference on Computer Vision and Pattern Recognition*, 2015, pp. 437–446.

[20] 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 IEEE Conference on Computer Vision and Pattern Recognition*, 2015, pp. 3287–3295.

[21] 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.

[22] 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.

[23] J. Hui, “mAP (mean Average Precision) for object detection,” Online at https://medium.com/@jonathan_hui/map-mean-average-precision-for-object-detection-45c121a31173.

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

[25] 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.

[26] 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.

[27] 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.

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

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

[30] 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

[31] 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.

[32] 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.

[33] 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.

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

[35] W. G. Najm, J. D. Smith, M. Yanagisawa *et al.*, “Pre-crash scenario typology for crash avoidance research,” United States. National Highway Traffic Safety Administration, Tech. Rep., 2007.

[36] 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.

[37] 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.

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

[39] 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.

[40] 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.

[41] 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 German Conference on Pattern Recognition*, 2014, pp. 31–42.

[42] J. L. Schönberger and J.-M. Frahm, “Structure-from-motion revisited,” in *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition*, 2016, pp. 4104–4113.

[43] R. Schubert, E. Richter, and G. Wanielik, “Comparison and evaluation of advanced motion models for vehicle tracking,” in *2008 11th International Conference on Information Fusion*. IEEE, 2008, pp. 1–6.

[44] R. Stiefelhagen, K. Bernardin, R. Bowers, J. Garofolo, D. Mostefa, and P. Soundararajan, “The CLEAR 2006 evaluation,” in *International evaluation workshop on classification of events, activities and relationships*. Springer, 2006, pp. 1–44.

[45] 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 IEEE Conference on Computer Vision and Pattern Recognition*, 2017, pp. 5038–5047.

[46] 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.

[47] T. Zhou, M. Brown, N. Snavely, and D. G. Lowe, “Unsupervised learning of depth and ego-motion from video,” in *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition*, 2017, pp. 6612–6619.