Long-range Multi-Object Tracking at Traffic Intersections on Low-Power Devices

Patrick Emami, Lily Elefteriadou, Sanjay Ranka

***Abstract*—The next generation of intelligent traffic signal control systems needs multi-object tracking (MOT) algorithms that can track vehicles hundreds of meters away from traffic intersections. To facilitate the integration of long-range MOT into existing traffic infrastructure, the tracker must achieve a good balance of cost-effectiveness, accuracy, and efficiency. Although much progress has been made on deep-learning-based MOT for video, these approaches have limited applicability for edge deployment since deep neural networks typically require power-hungry hardware accelerators to achieve real-time performance. Furthermore, traffic cameras have a field of view limited to near the intersection. To address these shortcomings, we introduce a practical MOT framework that fuses tracks from a novel video MOT neural architecture designed for low-power edge devices with tracks from a commercially available traffic radar. The proposed neural architecture achieves high efficiency by using depthwise separable convolutions to jointly predict object detections alongside a dense grid of features at a single scale for spatiotemporal object re-identification. A simple and effective late fusion strategy is also presented where tracks of distant vehicles from a traffic radar are handed over to the video tracker within a region where the sensor fields of view overlap. Our video tracker is empirically validated on the UA-DETRAC video MOT benchmark for traffic intersections and the multi-sensor tracker is evaluated on video and radar data collected and labeled by the authors at an instrumented traffic intersection.**

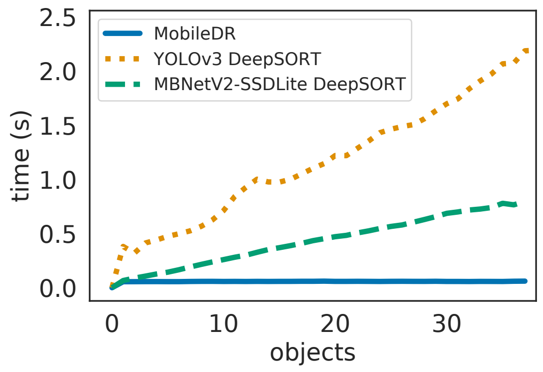


Figure 1: Time to predict object bounding boxes and Re-ID features for a single video frame as the number of detected objects increases *while running on CPUs*. We compare the proposed joint detection and tracking CNN, MobileDR, with YOLOv3 DeepSORT and MobileNetV2-SSDLite DeepSORT. DeepSORT's Re-ID CNN crops the image around each bounding box and processes each crop individually, which causes its runtime to depend linearly on the number of detected objects. MobileDR predicts Re-ID features in parallel for all objects with a single forward pass.

I. Introduction

INTELLIGENT traffic intersection control automates intersection management and improves adaptability to changing traffic conditions. These systems assume that a centralized intersection controller has access to detailed information about all traffic participants around the intersection in real time which it uses to make decisions about upcoming signal timings and autonomous vehicle trajectories [1]–[6]. It is expected that this will directly help reduce emissions, congestion, and accidents. However, to properly operate, information about vehicles that are still far down the road is needed. Early detection and tracking of vehicles gives the system sufficient time to properly optimize [6]. However, current multi-object tracking (MOT) algorithms for traffic intersections are by and large incapable of supporting this application, since they are either too inaccurate, too over-costed, or too inefficient for real-world applications. To achieve our high-level goal of introducing a practical framework for multi-object tracking at traffic intersections, we propose a novel convolutional neural network (CNN) for video MOT on low-power edge devices and a simple fusion algorithm for multi-sensor tracking with traffic camera and traffic radar.

State-of-the-art video MOT relies on computationally intensive CNNs that hardly run in real-time even when using a hardware accelerator (e.g., a GPU).[1] Accelerators that are powerful enough to run advanced CNNs have stringent power and cooling requirements and must be protected from adverse weather conditions. In most cases, such accelerators will not be available at the edge. Furthermore, traffic intersection control requires accurate tracking in regions far (e.g., 200+ meters) from the intersection, which is beyond the visual field of traffic cameras. Alternative solutions are based on classic computer vision techniques or other modalities like loop detectors and radar, which cannot provide sufficiently accurate results.

In this work, we introduce the MobileDR (**Mobile**NetV2 **D**etection and **R**e-ID) CNN architecture for video MOT on low-power edge devices. MobileDR jointly predicts object bounding boxes, class labels, and matchable features for object re-identification (Re-ID). Re-ID features are predicted as a dense grid at a single resolution for efficient parallel

P. Emami and S. Ranka are with the Department of Computer & Information Science & Engineering, University of Florida, Gainesville, FL. E-mail: pemami@ufl.edu, ranka@cise.ufl.edu. L. Elefteriadou is with the Department of Civil Engineering, University of Florida, Gainesville, FL. E-mail: elefter@ce.ufl.edu. Code: https://github.com/pemami4911/MobileDR

[1]To be concrete, assume "real-time" means the ability to process 10 video frames per second (FPS) or faster and "low-power" refers to achieving roughly 10 FPS running on hardware that consumes $5-10$ W.

extraction of object-centric features with a single forward pass (Figure 1). In our experiments, we study the trade-off of speed vs. performance induced by its lightweight architecture design.

To track vehicles far down the road, we propose a simple algorithm for combining video tracks with tracks provided by a traffic radar. The algorithm is based on a *late fusion* strategy because our approach is designed to require minimal effort for deployment at a traffic intersection by leveraging affordable and commercially available traffic radars. These sensors are typically shipped with proprietary tracking algorithms and provide an API for processing a real-time list of tracked objects. Therefore, we associate and fuse radar outputs with video at the level of tracks, as opposed to using a *mid-level fusion* strategy which would require access to the raw radar detections, or an *early fusion* strategy which would require access to raw acoustic waveforms from which to extract features to combine with video features for joint detection and association. Our fusion algorithm is designed around a "hand-off" of radar tracks of distant objects to video tracks of nearby objects within a region of overlap between the two sensor's field of views (Figures 2, 3). This is motivated by the observation that traffic radars have a much longer range than traffic cameras which creates a large region where the two sensors do not overlap. We also found that the radar was not well-suited for tracking objects near the intersection relative to video-based tracking, particularly in moderate-to-heavy congestion.

Our experiments analyze the performance of MobileDR on video-based detection and MOT as well as the performance of the multi-sensor MOT framework at an instrumented traffic intersection. With the same CPU hardware, MobileDR achieves a 450% increase in FPS over the popular YOLOv3 DeepSORT (3 vs. 20 FPS). We show that a dense prediction of Re-ID features removes a linear dependence of the runtime on the number of detected objects, regardless of the choice of detector (Figure 1). When augmenting MobileDR with traffic radar, we demonstrate a $+9.8\%$ improvement in MOTA, with most gains coming from improved tracking of vehicles far away from the intersection. To summarize, our main contributions are:

- We propose a novel joint detection and tracking CNN, MobileDR, that should be generally useful for real-time MOT on low-power edge devices.
- We introduce a multi-sensor MOT framework for traffic intersections that balances cost-effectiveness, accuracy, and efficiency, making it practical.
- We provide a thorough real-world evaluation on data collected from an instrumented traffic intersection.

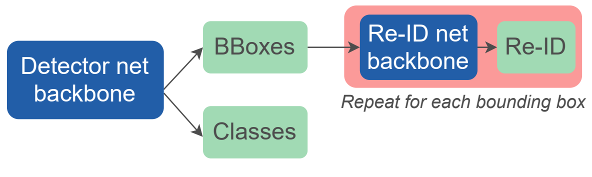
II. Related work

A. (Near-)real-time video MOT for traffic surveillance

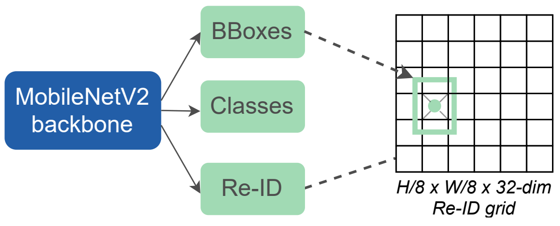
Hand-crafted pipelines: Methods that run efficiently on low-cost hardware have typically been based on hand-crafted image processing pipelines. Key steps include background subtraction, object segmentation, and motion estimation. See Wang et al. [7] for an overview. However, it is known that these methods degrade in the presence of variable illumination, cluttered environments with stop-and-go traffic, and diverse interacting objects with distinct motion patterns [8]–[12]. In general, the accuracy attainable by these methods is much worse than their deep learning counterparts.

Tracking-by-detection: The advent of powerful object detectors such as the Deformable Parts Model (DPM) [13], [14] ushered in the *tracking-by-detection* MOT paradigm. These methods first detect objects in each video frame and then match them over time to form tracks. State of the art tracking-by-detection methods use CNNs, which are invariant to translations in object location and are able to learn features that are robust to changes in illumination, shape, and color. However, the increase in detection accuracy achieved by CNNs comes at the price of requiring access to a hardware accelerator such as a GPU for deployment. For example, the SORT [15] algorithm is a simple and highly effective tracker that combines a Kalman Filter [16], the Munkres linear assignment algorithm [17] with a bounding box intersection-over-union (IOU) cost, and a powerful CNN for object detection. Trackers that use the IOU between bounding boxes in adjacent frames as a similarity measure for matching can be quite effective. V-IOU [18], which introduces a heuristic for handling missing bounding boxes in IOU-based tracking, is currently one of the leading methods on the UA-DETRAC traffic intersection MOT benchmark [19]. DeepSORT [20] improves over SORT by adding a metric-learning CNN for learning appearance features that are used in a cascaded spatiotemporal data association algorithm. Improvements to DeepSORT have since been proposed to further boost performance such as filtering out tracks with low confidence [21]. The video tracker described in this paper can be seen as an alternative to DeepSORT that can run in real time *without* needing a cumbersome hardware accelerator at the edge. CenterTrack [22] extends the powerful and efficient CenterNet [23] point-wise object detection framework for MOT. The key idea of CenterTrack is to unify detection and tracking into a single CNN where objects are treated as points and tracking is handled implicitly by predicting 2D offsets between points in *adjacent* frames. Data association is performed by a simple greedy matching on the predicted points. CenterTrack achieves strong performance on standard MOT benchmarks while processing 960×544 images in 57 ms on a GPU.

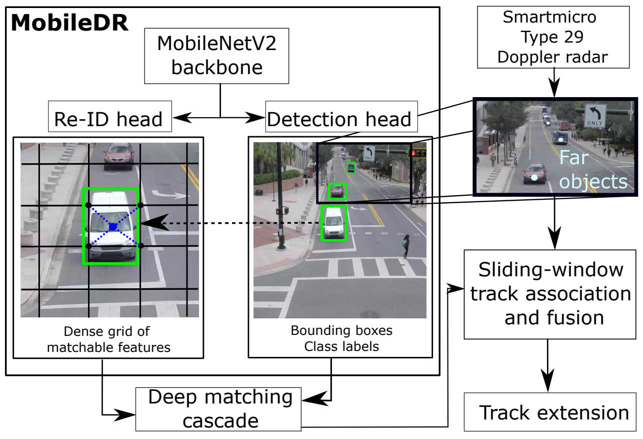
Joint detection and tracking: The joint detection and tracking (JDT) MOT paradigm has recently gained in interest because it aims to achieve faster FPS by combining the different deep neural networks used by tracking-by-detection trackers into a single CNN architecture. The basic idea is to attach an object detection head and a Re-ID head to a single CNN backbone and train with a multi-task loss. The detection head outputs one or more of: bounding boxes, object centers, object masks, or class scores. The Re-ID head predicts matchable features for data association. Recently proposed JDT trackers are Track R-CNN [24], RetinaTrack [25], JDE [26], UMA [27], and FairMOT [28]. Both our tracker, MobileDR, and FairMOT predict a dense grid of Re-ID features in parallel for all objects, unlike the other JDT methods. Like CenterTrack, FairMOT [28] also uses an anchor-less CenterNet detector as its backbone; however, CenterTrack does not *explicitly* predict Re-ID features for data association. FairMOT achieves state-of-the-art performance on standard MOT benchmarks by



(a) Tracking-by-detection



(b) MobileDR (Ours)



MobileDR
MobileNetV2 backbone
Re-ID head
Detection head
Smartmicro Type 29 Doppler radar
Far objects
Dense grid of matchable features
Bounding boxes Class labels
Sliding-window track association and fusion
Deep matching cascade
Track extension

(c)

Figure 2: a) Tracking-by-detection predicts deep Re-ID features for objects individually, which scales *linearly* with the number of detected objects. b) MobileDR is a simple and efficient CNN for joint detection and tracking MOT on low-power edge devices. It predicts a dense grid of Re-ID features at a single resolution. Matchable features for data association are bi-linearly interpolated from the centers of the non-maximally suppressed set of predicted bounding boxes. c) The multi-sensor tracking framework for long-range tracking at traffic intersections.

also introducing better pre-training strategies and optimization tricks for balancing detection and Re-ID objectives. Differently from FairMOT, we use MobileNetV2-SSDLite as the detection backbone, design the Re-ID head using only separable convolutions for maximum efficiency on edge processers, and train the Re-ID head independently of the detection branch with a supervised contrastive loss. Another related tracker is TubeTK [29], which proposes a novel approach to end-to-end MOT by regressing bounding box tubes from videos. So-called BTubes provide better spatiotemporal features and robustness to occlusion although the tracker itself is more heavy-weight as it requires using a sliding-window-based 3D CNN for processing video clips.

MOT at traffic intersections: Some of the aforementioned trackers have been evaluated on the task of traffic surveillance at intersections in prior work. The COSMOS [30] project curated a dataset of birds-eye-view 1920×1080 resolution video of an intersection and compared various “real-time” video trackers, varying the detector (Mask R-CNN [31], YOLOv3 [32], and SSD [33]) as well as the tracker (DAN [34], DeepSORT [20], and MCUT [30]). Mask R-CNN outperforms the other detectors but is roughly ten times slower on a GPU. DeepSORT is the fastest tracker, followed by DAN (37% slower).

B. Real-time multi-sensor MOT for traffic surveillance

Given that single-sensor traffic surveillance is limited by the choice of sensor, multi-sensor systems have been proposed to overcome this that aim to fuse information from multiple sensors. In one instance, video MOT and Doppler radar are fused for tracking oncoming vehicles at a traffic intersection [35]. The video solution is based on simple blob tracking whereas our method uses deep learning, and video and radar measurements are combined with early fusion to form a single set of tracks whereas we use late fusion. A system for parking lot and intersection monitoring was recently introduced [36] that is based on multi-camera tracking. They equip a group of networked cameras with power-efficient Jetson Nano GPUs to process video frames in real-time at the edge with a lightweight CNN. However, their system is designed for multi-camera tracking using tracking-by-detection. They avoid solving the challenge of real-time tracking-by-detection by simply not using CNN features for Re-ID. This lowers the overall accuracy of the system. Another system fuses multiple LiDAR sensors [37] placed around an intersection to track objects within the intersection region with reasonably high accuracy. In our work, we are interested in tracking vehicles *before* they arrive at the intersection; the limited range of LiDARs make them less suitable for this problem. To the best of our knowledge, no prior work exists that combines deep-learning-based video MOT with radar for traffic intersection surveillance.

III. Methodology

A. MobileDR

Backbone: For video MOT, we follow the JDT paradigm which achieves competitive performance compared to state-of-the-art tracking-by-detection methods while being orders of magnitude more efficient [24]–[26], [28]. We adapt the MobileNetV2-SSDLite [38] detector to create MobileDR, which has *three* output heads: a regression head for bounding boxes, an object classification head, and a novel fully-convolutional Re-ID head for object features (Figure 2b). Our motivation for using a MobileNet CNN backbone is that the MobileNet CNN family represents the state-of-the-art in accurate object detection on *low-power* devices [38]–[40], i.e., they are capable of running at real-time speeds on computing devices requiring less than ~ 10 W. The most recent iteration is MobileNetV3-SSDLite [40]; it achieves similar accuracy as V2 with a 27% improvement in latency at the cost of increased architecture complexity, so we use V2 in this work for simplicity and because V2 already achieves sufficiently low latency for our purposes. MobileNetV2 uses various architecture optimizations to help it run efficiently on edge devices. The most important one is *depthwise separable convolution* [39], [41], [42]. Essentially, this is a 2D convolution that has been factorized into two simpler operations. We note that each separable convolution uses BatchNorm [43] and ReLU activations. The entire architecture is a stack of inverted residual blocks (IRBs), which consist of a depthwise separable convolution and a linear residual bottleneck. The linearity of the bottleneck layer is important for preserving information, and the residual connection helps propagate gradients across multiple layers.

The regression and classification heads in SSDLite are defined similarly as in the SSD [33] except that they use separable convolutions. We follow the standard approach of attaching one set of detection heads after downsampling by $16\times$ as well as a head after each extra IRB at $32\times$ downsampling.

Re-ID head: In tracking-by-detection, a dedicated CNN is trained to produce a feature vector for each bounding box. This requires processing each bounding box individually which cannot typically be done in real-time if high-end compute resources are unavailable. For example, suppose 50 objects are detected in an image. Then, for each of the 50 bounding boxes, the image is cropped and the resulting patch is passed to the dedicated feature-extraction CNN. This fails to exploit the fact that most of the 50 image crops are overlapping. To take advantage of this, instead the bounding boxes can be processed "in parallel" by having the CNN produce a single dense grid of features after one forward pass. With this grid, for each bounding box we can extract an object feature by bi-linearly interpolating the feature at the center of the bounding box (Figure 2c). This removes the dependence of the runtime on the number of detected objects (Figure 1). Features are extracted from bounding box centers after running non-maximum suppression, which is superior to predicting an embedding for all positive bounding box anchorst [28]. The latter has issues when multiple anchors correspond to the same objec.

The Re-ID head consists of seven depthwise separable convolutions with 3×3 kernels and channel dimension progression $32 \rightarrow 256 \rightarrow 512 \rightarrow 1024 \rightarrow 512 \rightarrow 256 \rightarrow D$, where D is the output Re-ID feature dimension. The Re-ID head is fully-convolutional as it computes a dense grid without using any fully-connected layers (Figure 2b). Each grid coordinate (i, j) corresponds to a feature vector of dimension D. MobileDR estimates a single grid at $1/8\times$ the input resolution, attached after the third IRB. This resolution is high enough so that the model can predict distinct features for each object in a potentially highly cluttered scene. It has been previously demonstrated that deep feature matching performs best with features taken from intermediate CNN layers that have a large receptive field while still containing discriminative information [44]. We considered feature dimensions of $D = \{32, 64, 128\}$ and found 32 to work best based on a sensitivity analysis in our experiments.

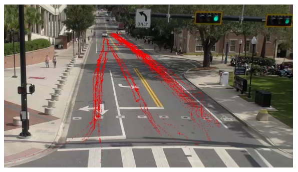
Training details: Jointly balancing losses for detection and Re-ID is nontrivial because the architecture design is asymmetric and can lead the model to favor one task over another [28]. To address this, we train MobileDR following a curriculum. First, we fine-tune a pretrained MobileNetV2-SSDLite model on the detection task using the standard SSD MultiBox training loss for bounding box regression and classification [33]. Then, we freeze the weights for the base layers, regression head, and classification head and train the randomly initialized weights of the Re-ID head using supervised contrastive learning.

The loss we use for the Re-ID head is the normalized temperature-scaled cross entropy loss, or NT-Xent loss [45]. This loss function encourages features corresponding to the same object (positive examples) to have a higher cosine similarity than features from two different objects (negative examples), and has been demonstrated to be highly successful for representation learning despite its simplicity. A key motivating factor for using this loss is that it does not rely on online mining of hard negatives, which is computationally expensive. To obtain positive and negative examples for evaluating the loss, we obtain them "for free" from track annotations using a manually-labeled MOT dataset. In detail, the NT-Xent loss is

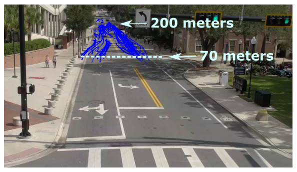
$$\begin{aligned}\mathcal{L} = &-\frac{1}{N}\sum_{i=1}^{N}\log\frac{\exp(\mathrm{CS}(x_i^t, x_i^{t'})/\tau)}{\sum_{k=1}^{N} I_{[k\neq i]}\exp(\mathrm{CS}(x_i^t, x_k^{t'})/\tau)}\\ &-\frac{1}{N}\sum_{i=1}^{N}\log\frac{\exp(\mathrm{CS}(x_i^{t'}, x_i^{t})/\tau)}{\sum_{k=1}^{N} I_{[k\neq i]}\exp(\mathrm{CS}(x_i^{t'}, x_k^{t})/\tau)},\end{aligned} \tag{1}$$

with CS := cosine similarity. The $I_{[k\neq i]}$ is an indicator variable equal to one when $k \neq i$ else it is zero. The numerators consist of the interpolated point-wise features from a mini-batch of N randomly sampled pairs of ground truth bounding boxes $\{(x_1^t, x_1^{t'}), \ldots, (x_N^t, x_N^{t'})\}$ originating from the same object in a video at frames $t < t'$. The denominators contain the negative pairs, which we implement by aligning each bounding box from one object in the mini-batch with a different object's bounding box. The temperature τ regularizes the Re-ID head parameters, helping to prevent them from getting stuck in poor local minima early on in training. To further improve generalization we use random mirroring and photometric data augmentation during training. During the initial detection training phase we use photometric distortion, random mirroring, and random image cropping.

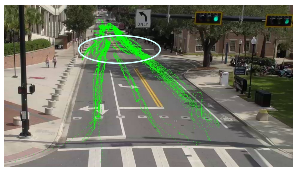
Data association: For each incoming video frame (e.g., of size 960×540), we first resize it to 300×300 and then use MobileDR to jointly predict bounding boxes, class labels, and the dense Re-ID feature grid. After applying hard non-maximum suppression to the bounding boxes based on class confidence scores and extracting features from the dense grid with bi-linear interpolation, the bounding boxes and features are sent to the deep matching cascade algorithm from DeepSORT [20]. We replace the features from the patch-based CNN with those extracted from the Re-ID grid, greatly increasing the tracker speed (Figure 1); see Wojke et al. [20] for full details of the matching algorithm. The MobileDR video tracker can be thought of as an alternative to DeepSORT that is suitable for real-time use on low-power edge devices.



(a) MobileDR



(b) Traffic radar



(c) MobileDR+Radar

Figure 3: The multi-sensor tracker obtains tracks from MobileDR (3a) and tracks from a traffic radar (3b). We keep radar tracks farther than a distance $\rho = 70$ meters from the intersection (shown as a dotted line) where the radar is most reliable. 3c) The multi-sensor tracker fuses radar and video tracks in the region of overlap (circled) to achieve efficient and high-quality tracking both near and far (up to 200 meters).

B. *MobileDR+Radar*

Due to the limited visual range of traffic cameras and our restriction of designing a low-cost and real-time solution, we propose to make use of off-the-shelf traffic radar to track vehicles far from traffic intersections. The late fusion strategy we describe next allows us to make use of affordable and widely available traffic radars that are easy to deploy at traffic intersections.

Fusion algorithm: Doppler-based traffic radar uses acoustic beams to detect moving objects and is able to estimate range and speed of vehicles at upwards of 200 meters from a traffic intersection. Off-the-shelf traffic radar units are often shipped with proprietary tracking algorithms and an API for accessing a real-time list of tracked objects. However, the tracking accuracy degrades for vehicles near the sensor if the angle of incidence between the moving object and the beam is not zero (known as the “cosine effect” [46]).[2] Also, we observed that the radar can generate a high number of false positive tracks from spurious detections due to clutter, particularly when there is moderate-to-heavy congestion. We handle this by removing radar tracks closer than a pre-set threshold of meters away from the intersection, leaving a stretch of road where the fields of view between the video and radar overlap (Figure 3). In this region, the algorithm identifies which video and radar tracklets originated from the same object and fuses the matched pairs.

An illustration of the full multi-sensor MOT framework is shown in Figure 2c. The key steps are summarized as follows:

1) Predict bounding boxes, class scores, and the dense grid of Re-ID features from the video frame at the current time step with MobileDR.
2) Bi-linearly interpolate object-centric features from the dense grid using the bounding boxes and after applying non-maximum suppression.
3) Update a video track array using the deep matching cascade data association algorithm.
4) Parse the latest set of radar tracks obtained from the real-time radar API.
5) Project the video and radar tracks into a sensor-centric world coordinate system with calibration matrix $\mathbf{P}$.
6) Do cross-sensor track-to-track association using relaxed linear assignment over the last w frames.
7) Fuse the matched tracks with simple averaging.
8) Extend a running estimate of the visible tracks in the field of view by solving a spatiotemporal assignment with the newly estimated fused tracks from the current sliding window.

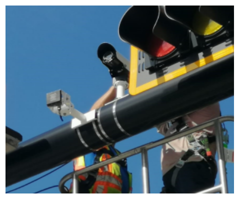
In detail, the algorithm gathers all video and radar tracklets within a time window of w seconds that are not deemed to be spurious. Spurious (i.e., false positive) tracklets are those that are too short-lived (less than m time steps long). Radar tracklets whose spatial centroid in world coordinates is closer than ρ meters to the intersection are considered spurious and removed (Figure 3b). Then, multi-sensor track association is performed by running a relaxed linear assignment. This measures the similarity between N video and M radar tracklets using the Euclidean dynamic time warping (DTW) distance [47]. The DTW is an efficient way to measure the similarity of 2D curves and is invariant to temporal stretching and shrinking. DTW provides an accurate tracklet similarity measurement as long as the sensors are calibrated such that tracks can be reasonably aligned in the world coordinate system; see Section III-C for a discussion on the calibration assumptions we make. We relax the one-to-one assignment constraint between the two sets of sensors tracklets, allowing for tracklets to go unmatched if the DTW distance is greater than a threshold. Threshold values are selected using a validation set of manually labeled video frames. In lieu of not having access to accurate track uncertainty information from commercial radar sensors, we handle tracklet fusion by averaging the positions and velocities of matched tracklets.

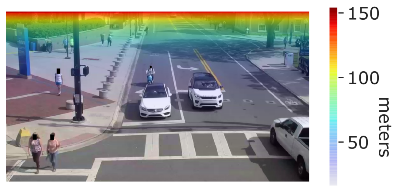
We maintain a set of tracks $\mathcal{T}$ which represents all objects believed to exist in the multi-sensor field of view up to but not including the current time window. A track extension algorithm based on relaxed linear assignment is used to update $\mathcal{T}$ with the new set of fused and unmatched tracklets. Similarity across time between candidate pairs of historical tracks and new tracklets is estimated by the pixel-space Euclidean distance between temporally adjacent pixel-space object states. We provide detailed pseudocode and more implementation details for the multi-sensor tracking algorithm in the appendix.

C. *Sensor installation and calibration*

To facilitate the calibration process and ensure sufficient overlap of field of view, we assume that the sensors are mounted on the same traffic mast arm as close to each other as possible and oriented in the same direction. We visualize the installation at the testbed for our experiments in Figure 4a.

[2]While multi-beam and continuous wave radar have also become commercially available—which in theory helps to address this—in this work we assume a standard single beam Doppler radar. We believe that more advanced radars can be used to achieve higher accuracy but at an increased cost.


(a)



(b)

Figure 4: a) Multi-sensor mounting configuration. b) Inspection of the estimated camera calibration matrix $\mathbf{P}$ by visualizing world space distances from the camera for each pixel.

Techniques for estimating the camera calibration matrix $\mathbf{P}$, which projects video tracks from the 2D image plane to a 3D world coordinate system, have been well-studied for traffic monitoring applications [48]. Necessary measurements of quantities such as camera height and line distances are carried out during the sensor installation process. In this work, we measured the length of a few lines visible in the camera image to obtain $\mathbf{P}$. See Figure 4b for a qualitative visualization of the depth estimation provided by $\mathbf{P}$. The software that came with our commercial radar sensor provided a tool for similarly calibrating the radar at the time of installation to enable extracting tracks in a world coordinate system. Since the sensors are installed at nearly the same position, we assume that the origins of both world coordinate systems are the same (that is, the sensor installation location on the mast arm). We did not find that this assumption resulted in inaccuracies being introduced into the multi-sensor tracking in our experiments.

IV. Experiments

In this section we evaluate MobileDR and MobileDR+Radar on data from various traffic intersections. Datasets, evaluation metrics, and training details for evaluating MobileDR are provided in Sections IV-A and IV-B. In Sections IV-C and IV-D, we show that MobileDR achieves strong detection and tracking precision, but due to its efficiency vs. performance trade-off, it has lower recall than other less efficient state-of-the-art architectures. Results from evaluating MobileDR+Radar on video and radar data collected by the authors at a live traffic intersection are in Section IV-E, where we verify that adding the radar sensor helps recover missed tracks, particularly in regions far from the intersection.

A. UA-DETRAC dataset and evaluation metrics

For assessing the performance of MobileDR we use the UA-DETRAC [19] video detection and tracking benchmark which consists of videos captured at 24 different intersections in China. These video sequences vary in time of day, weather, traffic level, and amount of occlusion. For the initial training stage (object detection), we use the UA-DETRAC V3 train, validation, and test splits. For the second training stage (Re-ID), we randomly sample $50,000$ pairs of bounding boxes from tracks in the training sequences and $5,000$ pairs from tracks in the validation sequences.

For object detection, we follow the UA-DETRAC benchmark and evaluate MobileDR using average precision (AP). We use the implementation of the AP metric as provided in the official evaluation toolkit, which computes the area under the precision-recall curve with the trapezoidal rule and linear interpolation. For MOT, again we follow the UA-DETRAC benchmark and evaluate the tracking performance using the PR CLEAR MOT metrics [60]. These metrics are similar to the CLEAR MOT [61] metrics except they aggregate results at various detection thresholds. The two most important metrics are the PR-MOTA and PR-IDF1 [62] scores. The PR-MOTA decreases when the number of false positives, false negatives, or fragmentations increases and hence is a good indicator of the strength of the *detector*. The PR-IDF1 metric measures the ability of the tracker to properly identify detections, and balances tracker precision and recall using the harmonic mean; hence, it is a good indicator of the strength of the Re-ID features and data association algorithm.

B. Training details

To train MobileDR on the detection task with the UA-DETRAC V3 data, we initialize the MobileNetV2 and SSDLite layers with parameters from a pretrained model implemented in PyTorch [63][3]. We use stochastic gradient descent with a learning rate of 0.01 for the heads and 0.001 for the base layers, momentum set to 0.9, and weight decay set to 0.0004. We multiply the learning rate by 0.1 after 80 epochs. The model is trained for 100 epochs with a batch size of 32 on a single NVIDIA Titan Xp GPU. We customize the SSDLite anchor boxes and aspect ratios by calculating average bounding box statistics with validation data from the UA-DETRAC dataset. The UA-DETRAC V3 classes are mapped to {background, bus, car, truck}.

When training the Re-ID head we freeze the parameters of the base layers and detection heads. This ensures that there will not be any detection performance degradation. We train the model for 20 epochs with a learning rate of 0.01 and batch size of 32. The loss temperature τ is initialized to one and gets divided by two every five epochs during training until it reaches 0.0625 where we keep it fixed thereafter.

C. UA-DETRAC detection

Key results: The AP scores on the UA-DETRAC test set are shown in Table I. We include published results listed on the benchmark's web-page for comparison—*notably, our model is the only one which can run in real time on a CPU.* The overall AP score of 58.17% is near the performance of the powerful Faster R-CNN [55] (58.45%). SpotNet [49] is the top-performing model which combines segmentation with the point-based detection approach of CenterNet [23].

Takeaways: MobileDR can detect vehicles relatively well when there is low amounts of occlusion and good illumination (as in the easy and medium test splits). As expected due to its lightweight design, it has more difficulty detecting partially occluded and small objects and handling low visibility due to poor weather. We attribute this to the fact that MobileDR has significantly fewer model parameters (Table III) than other more

[3]https://github.com/qfgaohao/pytorch-ssd

Table I: Image object detection results on the UA-DETRAC test set. As comparison we show published work appearing on the UA-DETRAC website (October 2020). MobileDR achieves comparable average precision (higher is better) to Faster R-CNN while being the only listed model to run in real-time on low-power edge devices.

Model	Real-time @ edge	Overall	Easy	Medium	Hard	Cloudy	Night	Rainy	Sunny
SpotNet [49]		86.80%	97.58%	92.57%	76.58%	89.38%	89.53%	80.93%	91.42%
FG-BR_Net [50]		79.96%	93.49%	83.60%	70.78%	87.36%	78.42%	70.50%	89.8%
HAT [51]		78.64%	93.44%	83.09%	68.04%	86.27%	78.00%	67.97%	88.78%
GP-FRCNNm [52]		77.96%	92.74%	82.39%	67.22%	83.23%	77.75%	70.17%	86.56%
R-FCN [53]		69.87%	93.32%	75.67%	54.31%	74.38%	75.09%	56.21%	84.08%
EB [54]		67.96%	89.65%	73.12%	53.64%	72.42%	73.93%	53.40%	83.73%
Faster R-CNN [55]		58.45%	82.75%	63.05%	44.25%	66.29%	69.85%	45.16%	62.34%
YOLOv2 [56]		57.72%	83.28%	62.25%	42.44%	57.97%	64.53%	47.84%	69.75%
RN-D [57]		54.69%	80.98%	59.13%	39.23%	59.88%	54.62%	41.11%	77.53%
MobileDR	✓	58.17%	86.03%	63.85%	41.39%	62.59%	66.50%	45.50%	64.00%

Table II: Video multi-object tracking results on the UA-DETRAC test set. As comparison we show published work appearing on the UA-DETRAC website (October 2020).

Model	Real-time @ edge	PR-MOTA (↑)	PR-MOTP (↓)	PR-MT (↑)	PR-ML (↓)	PR-IDS (↓)	PR-FRAG (↓)	PR-FP (↓)	PR-FN (↓)
Mask R-CNN+V-IOU [18]		30.7%	37.4%	28.7%	23.2%	143.3	1183.1	13387.9	195193.9
EB+DAN [34]		20.2%	26.3%	14.5%	18.1%	518.2	-	9747.8	135978.1
CompACT+FAMNet [58]		19.8%	36.7%	17.1%	18.2%	617.4	970.2	14988.6	164432.6
EB+IOUT [59]		19.4%	28.9%	17.7%	18.4%	2311.3	2445.9	14796.5	171806.8
MobileDR	✓	15.4%	30.4%	10.7%	30.1%	397.4	686.3	13937.1	255923.8

accurate and heavyweight detectors. While this may appear as a shortcoming of the MobileDR architecture, in fact it further justifies the use of additional sensors such as a Doppler radar to augment it in practice. We discuss ways to potentially improve its performance in future work in Section V.

D. UA-DETRAC tracking

Key results: Results for the UA-DETRAC MOT test set are in Table II. The baselines for comparison in the table are the published results from the public web server at the time of writing. We note that each entry uses its own detections, which greatly influences the tracking performance. The PR-MOTA achieved by MobileDR is 15.4% and the PR-MOTP is 30.4%. While this surpasses the PR-MOTP score of EB+IOUT by 1.5%, the PR-MOTA score is 4% lower. The high PR-MOTP score is due in part to the strength of the learned Re-ID features; notice the low number of ID switches (397.4, the second lowest), fragments (686.3, the lowest), and false positives (13937.1, third lowest). The relatively high number of false negatives (i.e., missed tracks) are chiefly caused by detection failures and explains the low PR-MOTA score.

Takeaways: For a qualitative look at the tracker performance see Figure 5. This helps illustrate that it successfully detects and tracks most vehicles that are fully or near-fully visible. Vehicles that are distant or mostly occluded cause the majority of tracking failures. The conclusion we draw from the tracking analysis is that although the MobileDR has high precision and is effective at tracking vehicles near the intersection, it tends to miss vehicles that are more challenging to track. This means that this tracker is well-suited to be integrated into a multi-sensor tracking framework that can complement the video tracking by providing missing track information for e.g., vehicles that are beyond the range of the video sensor.

E. MobileDR+Radar evaluation

Dataset and metrics: We mounted an Image Sensing Systems Autoscope traffic camera at an intersection in Gainesville, FL, which provides a stream of 720×1280 RGB frames at about 30 FPS to a traffic signal mast arm. The Doppler radar is a Smartmicro Systems Type 29 that sends a list of tracked objects with IDs at a rate of about 20 Hz to the traffic intersection cabinet from which it can be accessed. We collected and manually annotated $4,324$ video frames with track labels using the Vatic labeler [64]. We stored the stream of radar tracks during this time period as well. The video sequence is split into three shorter sequences: an *easy* sequence (light traffic) which is frames 1–$1,830$, a *hard* sequence (heavy traffic) which is frames $1,830$–$3,277$, and we set aside frames $3,227$–$4,324$ as a tuning set for analyzing tracker hyperparameters. For certain experiments, we also separately analyzed model performance on all tracks whose location in a video frame was closer than $\rho = 70$ meters to the intersection (Near) and those tracks farther than $\rho = 70$ meters away (Far). *We do not train MobileDR on this data* and instead re-use the MobileDR model trained on

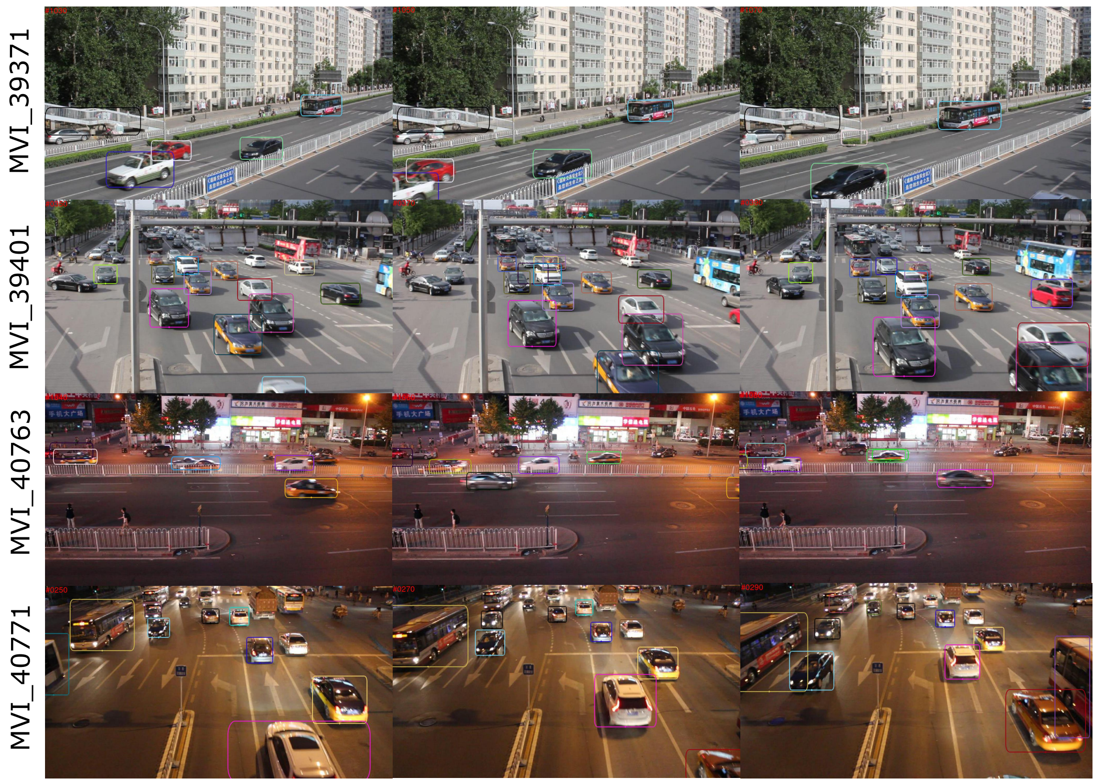


Figure 5: MobileDR tracking from the UA-DETRAC test set. Each column is twenty frames apart (left to right).

Table III: Video-only tracking efficiency comparison using a 6-core Intel i5-9600K CPU on the Gainesville validation sequence.

Tracker	Image size	# params (M)	CPU FPS (↑)	IDF1 (↑)	MOTA (↑)
MobileDR, 32-D	300×300	4.448	19.37	57.5%	38.5%
MobileDR, 64-D	300×300	4.456	19.31	56.6%	38.6%
MobileDR, 128-D	300×300	4.473	20.99	56.4%	37.5%
DeepSORT	320×320	73.442	3.51	55.0%	44.2%

Table IV: Ablations and parameter analysis for the multi-sensor tracker using the Gainesville validation sequence. We use $\rho = 70$ meters. Latency measures the time in seconds to run cross-sensor track association and fusion using w frames.

No radar < ρ	Window size (frames)	Near (< ρ) IDF1/MOTA	Far ($\geq \rho$) IDF1/MOTA	Both, IDF1/MOTA	Latency (↓)
	20	63.1%/41.3%	45.7%/34.0%	45.6%/36.4%	-
✓	10	74.4%/64.1%	41.7%/36.2%	41.8%/43.6%	0.071 s
✓	20	75.7%/65.2%	46.2%/35.5%	46.5%/43.3%	0.077 s
✓	30	73.9%/62.4%	52.9%/39.5%	**53.5%/45.5%**	0.072 s
✓	40	79.4%/63.6%	49.3%/38.2%	53.3%/45.2%	0.080 s

the larger UA-DETRAC dataset. This allows us to also explore the generalization abilities of MobileDR since the Gainesville data is recorded on a different continent and with a different traffic camera than the UA-DETRAC data. We rely on the standard CLEAR MOT metrics at a detection threshold of 0.3 for easy interpretation of the tracking results.

We compare MobileDR+Radar against DeepSORT, MobileDR (without radar), radar alone, and a simple baseline that replicates a classic real-time video surveillance tracking pipeline of Mixture-of-Gaussians (MoG) background subtraction followed by blob tracking. Blob tracking is accomplished by first running blob detection on the MoG foreground mask, then using a Kalman Filter to obtain a set of blob track hypotheses, and finally applying the Munkres [17] linear assignment algorithm for data association. This baseline is implemented with OpenCV [65] and represents a simple MOT algorithm that practitioners may attempt to use in absence of a GPU. See the appendix for more details.

Efficiency comparison: First, we compare the number of model parameters and tracking latency for MobileDR with Re-ID output dimensions $\{32, 64, 128\}$ against DeepSORT [20] in Table III. All of the evaluation is conducted on a single PC with 32 GB of RAM and a 6-core Intel Core i5-9600K CPU. MobileDR achieves a 450% increase (3.5 vs. 19.37) in FPS over DeepSORT and a 94% decrease in the number of model parameters. We note that when deploying MobileDR on an edge device as opposed to a PC, one should expect a slightly lower FPS. We use the 32-dim features for all other experiments since it has the smallest memory footprint.

Sensitivity analysis: For MobileDR+Radar, we analyze the impact of removing radar tracks closer than $\rho = 70$ meters to the intersection as well as the choice of window length w. By

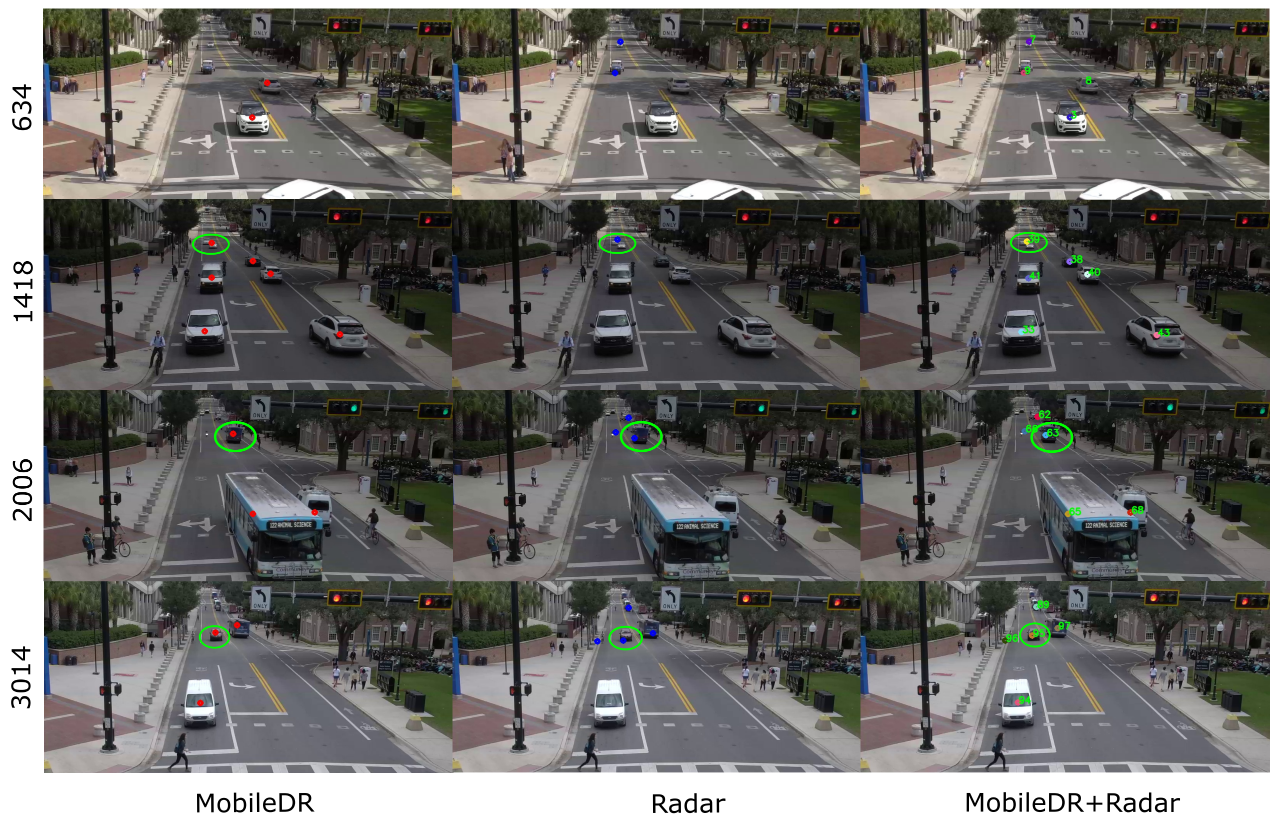


Figure 6: Tracking outputs for Easy sequence (frames 634, 1418) and Hard sequence (frames 2006, 3014). We highlight tracks that overlapped between sensors and were subsequently matched and fused. We show the IDs of the fused tracklets.

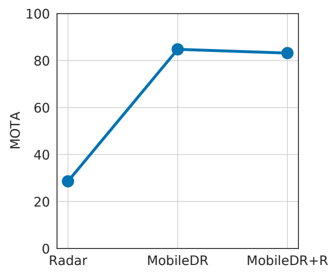
Table V: Gainesville dataset CLEAR MOT results (key metrics are highlighted). **Adding radar increases MobileDR's MOTA by $+9.8\%$ on the Hard sequence.**

	Tracker	MOTA (↑)	MOTP (↓)	IDF1 (↑)	MT	PT	ML	FP	FN	Rcll	Prcn	IDsw	Frag
Easy	MobileDR+Radar	46.4%	119.6	48.0%	3	12	1	1738	1778	73.8%	73.7%	43	58
	MobileDR	44.2%	66.3	60.2%	0	13	3	109	3589	45.9%	96.5%	5	14
	Radar only	24.9%	167.0	40.3%	2	13	1	2438	2502	62.3%	62.9%	42	68
	OpenCV blob tracking	−12.6%	239.5	3.6%	0	0	16	1019	6384	3.9%	19.8%	67	68
	DeepSORT[†]	67.3%	62.0	61.3%	5	10	1	580	1554	76.6%	89.8%	39	34
Hard	MobileDR+Radar	38.4%	133.2	38.2%	1	12	4	989	2520	56.3%	76.6%	42	81
	MobileDR	29.6%	58.2	46.8%	1	9	7	194	3864	33.0%	90.7%	1	6
	Radar only	19.3%	172.4	33.9%	0	13	4	1499	3109	46.1%	63.9%	43	88
	OpenCV blob tracking	−19.1%	317.6	3.9%	0	3	14	1297	5512	4.4%	16.2%	55	59
	DeepSORT[†]	45.3%	65.9	59.3%	3	9	5	420	2726	52.7%	87.9%	7	13

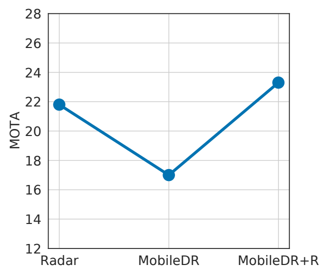
[†] Runs at 3.5 FPS on a CPU, see Table III

comparing the first ($\rho = 0$, $w = 20$) and third ($\rho = 70$, $w = 20$) rows of Table IV, we can see that $\rho = 0$ causes a sharp drop in Near ($< \rho$) IDF1 (63.1% vs. 75.7%) and Near ($< \rho$) MOTA (41.3% vs. 65.2%). Naturally, there is little change in Far ($\geq \rho$) IDF1 and MOTA scores. When we compare window lengths of $\{10, 20, 30, 40\}$ frames (equivalently $\{0.33, 0.67, 1, 1.33\}$ seconds for a 30 FPS camera), we see an improvement in IDF1 and MOTA as the window length increases with a noticeable jump occurring between $w = 20$ and $w = 30$. Since the difference in latency of the multi-sensor tracker for the best performing window size $w = 30$ compared to the other window sizes is negligible, we use $w = 30$ for the remainder of our experiments. Effectively, the algorithm only has to spend an additional 0.072 seconds to process each new batch of tracklets every w frames.

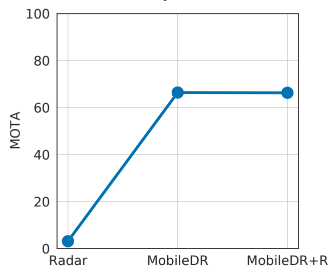
Key results: The tracking performance for the easy and hard Gainesville sequences are presented in Table V. We visualize examples of our method's track association and fusion in Figure 6. MobileDR+Radar outperforms MobileDR as well as the radar alone on the easy sequence. In Figures 7a and 7b, we directly compare the MOTA of these three trackers on the easy sequences as broken down by "near" and "far" based on ρ. MobileDR+Radar drastically improves the MOTA of MobileDR for the "far" region (Figure 7b). Note the increase in tracking recall from 45.9% to 73.8%. However, adding the radar decreases the MOTP because the radar tracking is less precise than the video tracking. Similarly, the lower overall IDF1 score of the fused solution compared to MobileDR is due to the less reliable radar sensor; however, notice that the fused solution's IDF1 scores are higher than the radar's IDF1 scores.



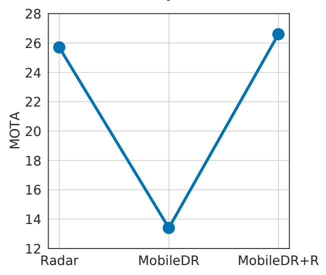
(a) Easy / Near



(b) Easy / Far



(c) Hard / Near



(d) Hard / Far

Figure 7: A comparison of CLEAR MOTA scores on the Gainesville dataset broken down by sequence difficulty and proximity to the intersection as measured by ρ. Higher is better. **When considering all four scenarios together, the fused solution is the highest performing.**

DeepSORT achieves strong results (MOTA of 67.3% and IDF1 of 61.3%—MobileDR's IDF1 score of 60.2% is similar) due to the use of the accurate YOLOv3 detector, but we recall that it requires a GPU to achieve real-time performance at the edge.

The OpenCV blob tracker has remarkably poor performance compared to the other trackers. First, it has no way to *classify* objects and has to designate any moving blob as a positive track. Second, the MoG background subtraction loses stationary objects such as vehicles that are stopped at red lights. Finally, blob detection and tracking requires applying morphological operations to remove spurious blobs, which leads to a high number of false negatives.

In the hard sequence, we see a large $+9.8\%$ improvement in overall MOTA when adding radar to MobileDR. DeepSORT struggles in denser traffic, and we see that the gap in MOTA between MobileDR+Radar and DeepSORT has shrunk compared to the easy sequence. MobileDR+Radar achieves a lower number of false negatives than DeepSORT and has a slightly higher tracking recall due to its ability to track vehicles far from the intersection. Unsurprisingly, the radar and blob tracker show a decline in overall tracking performance with denser traffic.

Takeaways: We demonstrated that MobileDR+Radar outperforms MobileDR and radar alone in terms of MOTA both near *and* far from the intersection in dense traffic. In constrained resource settings such as a real-world deployment of a intelligent traffic intersection control system, we believe MobileDR+Radar provides a reasonably good alternative to heavyweight trackers like YOLOv3 DeepSORT.

F. Failure modes

We explored fusing DeepSORT [20] with the Doppler radar on the Gainesville dataset to try to improve its tracking performance at far distances. We found that fusing DeepSORT with the radar *decreased* DeepSORT's performance as measured by the MOT metrics. The radar produces a high number of false positives and ID switches relative to DeepSORT, and despite that the fused result had higher recall (since vehicles far away from the intersection were tracked more often), the overall performance was lower. This implies that more powerful deep-learning-based trackers need to be paired with more advanced radars (e.g., multi-beam Doppler radar as opposed to the single beam radar used in this work) to see a benefit from the multi-sensor fusion.

Since MobileDR is trained with supervised learning using data from the UA-DETRAC training set, we observe drops in performance when deploying it without fine-tuning at new intersections, such as the one in Gainesville. For example, we notice an increase in spurious detections, which we believe can be attributed to a distribution shift induced by test images taken with a traffic camera that is positioned and oriented differently than the cameras used to create the training set. MobileDR struggled in dense traffic scenarios during which there were many vehicles far down the road from the intersection (e.g., Figure 7d). Since it is unreasonable to require collecting and annotating training data at every traffic intersection, new data augmentation strategies should be explored to help achieve stronger out-of-distribution generalization.

V. Discussion

In this work, we developed a practical multi-sensor MOT algorithm that balances cost-effectiveness, accuracy, and efficiency for downstream applications such as intelligent traffic intersection control. To that end, we introduced MobileDR, a CNN for real-time video MOT on low-power hardware that jointly predicts object detections and matchable object features for data association. MobileDR is combined with a deep matching cascade in the video tracker, which is then complemented by an off-the-shelf Doppler radar, resulting in clear improvements in tracking performance for objects that are far away from the traffic intersection.

We note that we could further push performance by upgrading the MobileDR's detection branch and base layers to MobileNetV3-SSDLite [40] to further reduce latency. Replacing the SSDLite heads with CenterNet [23] or SpotNet [49]-style detection (treating objects as points) should also improve accuracy because these methods are anchor-free and perform center-offset regression. In future work, we will integrate the tracker into a complete intelligent traffic intersection controller [6] and conduct field experiments to evaluate the entire system jointly.

Acknowledgment

The authors would like to thank Emmanuel Posadas and Chris Dolan from the City of Gainesville Public Works department for assistance with the traffic sensor installation. This work is supported in part by National Science Foundation

(NSF) grant awards 1446813 and 1922782 and Florida Dept. of Transportation (FDOT) grant award BDV31 TWO 977-116. Patrick Emami is also supported in part by a FEF McKnight fellowship and a UF CISE graduate research fellowship. The opinions, findings, and conclusions expressed in this publication are those of the author(s) and not necessarily those of the FDOT or of the NSF.

References

[1] Z. Li, L. Elefteriadou, and S. Ranka, "Signal control optimization for automated vehicles at isolated signalized intersections," *Transportation Research Part C: Emerging Technologies*, vol. 49, pp. 1–18, 2014.

[2] P. Lin, J. Liu, P. J. Jin, and B. Ran, "Autonomous vehicle-intersection coordination method in a connected vehicle environment," *IEEE Intelligent Transportation Systems Magazine*, vol. 9, no. 4, pp. 37–47, 2017.

[3] C. Yu, Y. Feng, H. X. Liu, W. Ma, and X. Yang, "Integrated optimization of traffic signals and vehicle trajectories at isolated urban intersections," *Transportation Research Part B: Methodological*, vol. 112, pp. 89–112, 2018.

[4] B. Xu, X. J. Ban, Y. Bian, W. Li, J. Wang, S. E. Li, and K. Li, "Cooperative method of traffic signal optimization and speed control of connected vehicles at isolated intersections," *IEEE Transactions on Intelligent Transportation Systems*, vol. 20, no. 4, pp. 1390–1403, 2018.

[5] Y. Guo, J. Ma, C. Xiong, X. Li, F. Zhou, and W. Hao, "Joint optimization of vehicle trajectories and intersection controllers with connected automated vehicles: Combined dynamic programming and shooting heuristic approach," *Transportation research part C: emerging technologies*, vol. 98, pp. 54–72, 2019.

[6] M. Pourmehrab, L. Elefteriadou, S. Ranka, and M. Martin-Gasulla, "Optimizing signalized intersections performance under conventional and automated vehicles traffic," *IEEE Transactions on Intelligent Transportation Systems*, vol. 21, no. 7, pp. 2864–2873, 2020.

[7] G. Wang, D. Xiao, and J. Gu, "Review on vehicle detection based on video for traffic surveillance," in *2008 IEEE International Conference on Automation and Logistics*. IEEE, 2008, pp. 2961–2966.

[8] N. Saunier and T. Sayed, "A feature-based tracking algorithm for vehicles in intersections," in *The 3rd Canadian Conference on Computer and Robot Vision (CRV'06)*. IEEE, 2006, pp. 59–59.

[9] S. Aslani and H. Mahdavi-Nasab, "Optical flow based moving object detection and tracking for traffic surveillance," *International Journal of Electrical, Computer, Energetic, Electronic and Communication Engineering*, vol. 7, no. 9, pp. 1252–1256, 2013.

[10] K. Kale, S. Pawar, and P. Dhulekar, "Moving object tracking using optical flow and motion vector estimation," in *2015 4th international conference on reliability, infocom technologies and optimization (ICRITO)(trends and future directions)*. IEEE, 2015, pp. 1–6.

[11] M. Chaple and S. Paygude, "Vehicle detection and tracking from video frame sequence," *International Journal of Scientific & Engineering Research*, vol. 4, no. 3, pp. 1–7, 2013.

[12] J. Jodoin, G. Bilodeau, and N. Saunier, "Tracking all road users at multimodal urban traffic intersections," *IEEE Transactions on Intelligent Transportation Systems*, vol. 17, no. 11, pp. 3241–3251, 2016.

[13] P. F. Felzenszwalb, R. B. Girshick, D. McAllester, and D. Ramanan, "Object detection with discriminatively trained part-based models," *IEEE transactions on pattern analysis and machine intelligence*, vol. 32, no. 9, pp. 1627–1645, 2009.

[14] P. F. Felzenszwalb, R. B. Girshick, and D. McAllester, "Cascade object detection with deformable part models," in *2010 IEEE Computer society conference on computer vision and pattern recognition*. IEEE, 2010, pp. 2241–2248.

[15] A. Bewley, Z. Ge, L. Ott, F. Ramos, and B. Upcroft, "Simple online and realtime tracking," in *2016 IEEE International Conference on Image Processing (ICIP)*. IEEE, 2016, pp. 3464–3468.

[16] R. E. Kalman, "A new approach to linear filtering and prediction problems," *Transaction of the ASME—Journal of Basic Engineering*, pp. 33–45, 1960.

[17] J. Munkres, "Algorithms for the assignment and transportation problems," *Journal of the society for industrial and applied mathematics*, vol. 5, no. 1, pp. 32–38, 1957.

[18] E. Bochinski, T. Senst, and T. Sikora, "Extending iou based multi-object tracking by visual information," in *2018 15th IEEE International Conference on Advanced Video and Signal Based Surveillance (AVSS)*. IEEE, 2018, pp. 1–6.

[19] S. Lyu, M. C. Chang, D. Du, W. Li, Y. Wei, M. D. Coco, P. Carcagni, A. Schumann, B. Munjal, D. Q. T. Dang, D. H. Choi, E. Bochinski, F. Galasso, F. Bunyak, G. Seetharaman, J. W. Baek, J. T. Lee, K. Palaniappan, K. T. Lim, K. Moon, K. J. Kim, L. Sommer, M. Brandlmaier, M. S. Kang, M. Jeon, N. M. Al-Shakarji, O. Acatay, P. K. Kim, S. Amin, T. Sikora, T. Dinh, T. Senst, V. G. H. Che, Y. C. Lim, Y. M. Song, and Y. S. Chung, "UA-DETRAC 2018: Report of AVSS2018 IWT4S Challenge on Advanced Traffic Monitoring," *Proceedings of AVSS 2018 - 2018 15th IEEE International Conference on Advanced Video and Signal-Based Surveillance*, 2019.

[20] N. Wojke, A. Bewley, and D. Paulus, "Simple online and realtime tracking with a deep association metric," in *2017 IEEE international conference on image processing (ICIP)*. IEEE, 2017, pp. 3645–3649.

[21] X. Hou, Y. Wang, and L.-P. Chau, "Vehicle tracking using deep sort with low confidence track filtering," in *2019 16th IEEE International Conference on Advanced Video and Signal Based Surveillance (AVSS)*. IEEE, 2019, pp. 1–6.

[22] X. Zhou, V. Koltun, and P. Krähenbühl, "Tracking objects as points," *arXiv preprint arXiv:2004.01177*, 2020.

[23] X. Zhou, D. Wang, and P. Krähenbühl, "Objects as points," *arXiv preprint arXiv:1904.07850*, 2019.

[24] P. Voigtlaender, M. Krause, A. Osep, J. Luiten, B. B. G. Sekar, A. Geiger, and B. Leibe, "Mots: Multi-object tracking and segmentation," jun 2019.

[25] Z. Lu, V. Rathod, R. Votel, and J. Huang, "Retinatrack: Online single stage joint detection and tracking," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2020, pp. 14 668–14 678.

[26] Z. Wang, L. Zheng, Y. Liu, Y. Li, and S. Wang, "Towards real-time multi-object tracking," *arXiv preprint arXiv:1909.12605*, 2019.

[27] J. Yin, W. Wang, Q. Meng, R. Yang, and J. Shen, "A unified object motion and affinity model for online multi-object tracking," in *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2020, pp. 6767–6776.

[28] Y. Zhang, C. Wang, X. Wang, W. Zeng, and W. Liu, "Fairmot: On the fairness of detection and re-identification in multiple object tracking," *arXiv preprint arXiv:2004.01888*, 2020.

[29] B. Pang, Y. Li, Y. Zhang, M. Li, and C. Lu, "Tubetk: Adopting tubes to track multi-object in a one-step training model," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2020, pp. 6308–6318.

[30] S. Yang, E. Bailey, Z. Yang, J. Ostrometzky, G. Zussman, I. Seskar, and Z. Kostic, "Cosmos smart intersection: Edge compute and communications for bird's eye object tracking," in *Proc. 4th International Workshop on Smart Edge Computing and Networking (SmartEdge'20)*, 2020.

[31] K. He, G. Gkioxari, P. Dollár, and R. Girshick, "Mask r-cnn," in *Proceedings of the IEEE international conference on computer vision*, 2017, pp. 2961–2969.

[32] J. Redmon and A. Farhadi, "Yolov3: An incremental improvement," *arXiv preprint arXiv:1804.02767*, 2018.

[33] W. Liu, D. Anguelov, D. Erhan, C. Szegedy, S. Reed, C.-Y. Fu, and A. C. Berg, "Ssd: Single shot multibox detector," in *European conference on computer vision*. Springer, 2016, pp. 21–37.

[34] S. Sun, N. Akhtar, H. Song, A. S. Mian, and M. Shah, "Deep affinity network for multiple object tracking," *IEEE transactions on pattern analysis and machine intelligence*, 2019.

[35] A. Roy, N. Gale, and L. Hong, "Automated traffic surveillance using fusion of doppler radar and video information," *Mathematical and Computer Modelling*, vol. 54, no. 1-2, pp. 531–543, 2011.

[36] M. Nikodem, M. Słabicki, T. Surmacz, P. Mrówka, and C. Dołęga, "Multi-camera vehicle tracking using edge computing and low-power communication," *Sensors*, vol. 20, no. 11, p. 3334, 2020.

[37] H. Zhao, J. Sha, Y. Zhao, J. Xi, J. Cui, H. Zha, and R. Shibasaki, "Detection and tracking of moving objects at intersections using a network of laser scanners," *IEEE transactions on intelligent transportation systems*, vol. 13, no. 2, pp. 655–670, 2011.

[38] M. Sandler, A. Howard, M. Zhu, A. Zhmoginov, and L.-C. Chen, "Mobilenetv2: Inverted residuals and linear bottlenecks," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2018, pp. 4510–4520.

[39] A. G. Howard, M. Zhu, B. Chen, D. Kalenichenko, W. Wang, T. Weyand, M. Andreetto, and H. Adam, "Mobilenets: Efficient convolutional neural networks for mobile vision applications," *arXiv e-prints*, Apr. 2017. [Online]. Available: https://ui.adsabs.harvard.edu/abs/2017arXiv170404861H

[40] A. Howard, M. Sandler, G. Chu, L.-C. Chen, B. Chen, M. Tan, W. Wang, Y. Zhu, R. Pang, V. Vasudevan *et al.*, "Searching for mobilenetv3," in

Proceedings of the IEEE International Conference on Computer Vision, 2019, pp. 1314–1324.
[41] F. Chollet, “Xception: Deep learning with depthwise separable convolutions,” jul 2017.
[42] X. Zhang, X. Zhou, M. Lin, and J. Sun, “Shufflenet: An extremely efficient convolutional neural network for mobile devices,” in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2018, pp. 6848–6856.
[43] S. Ioffe and C. Szegedy, “Batch normalization: Accelerating deep network training by reducing internal covariate shift,” *arXiv preprint arXiv:1502.03167*, 2015.
[44] J. L. Long, N. Zhang, and T. Darrell, “Do convnets learn correspondence?” in *Advances in neural information processing systems*, 2014, pp. 1601–1609.
[45] T. Chen, S. Kornblith, M. Norouzi, and G. Hinton, “A simple framework for contrastive learning of visual representations,” *arXiv preprint arXiv:2002.05709*, 2020.
[46] M. E. Goodson, “Technical shortcomings of doppler traffic radar,” *Journal of Forensic Science*, vol. 30, no. 4, pp. 1186–1193, 1985.
[47] S. Salvador and P. Chan, “Toward accurate dynamic time warping in linear time and space,” *Intelligent Data Analysis*, vol. 11, no. 5, pp. 561–580, 2007.
[48] N. K. Kanhere and S. T. Birchfield, “A taxonomy and analysis of camera calibration methods for traffic monitoring applications,” *IEEE Transactions on Intelligent Transportation Systems*, vol. 11, no. 2, pp. 441–452, 2010.
[49] H. Perreault, G.-A. Bilodeau, N. Saunier, and M. Héritier, “SpotNet: Self-Attention Multi-Task Network for Object Detection,” 2020. [Online]. Available: http://arxiv.org/abs/2002.05540
[50] Z. Fu, Y. Chen, H. Yong, R. Jiang, L. Zhang, and X.-S. Hua, “Foreground gating and background refining network for surveillance object detection,” *IEEE Transactions on Image Processing*, vol. 28, no. 12, pp. 6077–6090, Dec. 2019.
[51] S. Wu, M. Kan, S. Shan, and X. Chen, “Hierarchical attention for part-aware face detection,” *International Journal of Computer Vision*, vol. 127, no. 6-7, pp. 560–578, mar 2019.
[52] S. Amin and F. Galasso, “Geometric proposals for faster r-cnn,” in *2017 14th IEEE International Conference on Advanced Video and Signal Based Surveillance (AVSS)*. IEEE, 2017, pp. 1–6.
[53] J. Dai, Y. Li, K. He, and J. Sun, “R-fcn: Object detection via region-based fully convolutional networks,” in *Advances in neural information processing systems*, 2016, pp. 379–387.
[54] L. Wang, Y. Lu, H. Wang, Y. Zheng, H. Ye, and X. Xue, “Evolving boxes for fast vehicle detection,” in *2017 IEEE international conference on multimedia and Expo (ICME)*. IEEE, 2017, pp. 1135–1140.
[55] 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.
[56] J. Redmon and A. Farhadi, “Yolo9000: better, faster, stronger,” in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2017, pp. 7263–7271.
[57] H. Perreault, G.-A. Bilodeau, N. Saunier, and P. Gravel, “Road user detection in videos,” *arXiv preprint arXiv:1903.12049*, 2019.
[58] P. Chu and H. Ling, “Famnet: Joint learning of feature, affinity and multi-dimensional assignment for online multiple object tracking,” in *Proceedings of the IEEE International Conference on Computer Vision*, 2019, pp. 6172–6181.
[59] E. Bochinski, V. Eiselein, and T. Sikora, “High-speed tracking-by-detection without using image information,” in *2017 14th IEEE International Conference on Advanced Video and Signal Based Surveillance (AVSS)*. IEEE, 2017, pp. 1–6.
[60] L. Wen, D. Du, Z. Cai, Z. Lei, M.-C. Chang, H. Qi, J. Lim, M.-H. Yang, and S. Lyu, “Ua-detrac: A new benchmark and protocol for multi-object detection and tracking,” *arXiv preprint arXiv:1511.04136*, 2015.
[61] K. Bernardin, A. Elbs, and R. Stiefelhagen, “Multiple object tracking performance metrics and evaluation in a smart room environment,” in *Sixth IEEE International Workshop on Visual Surveillance, in conjunction with ECCV*, vol. 90. Citeseer, 2006, p. 91.
[62] E. Ristani, F. Solera, R. Zou, R. Cucchiara, and C. Tomasi, “Performance measures and a data set for multi-target, multi-camera tracking,” in *European Conference on Computer Vision*. Springer, 2016, pp. 17–35.
[63] A. Paszke, S. Gross, F. Massa, A. Lerer, J. Bradbury, G. Chanan, T. Killeen, Z. Lin, N. Gimelshein, L. Antiga *et al.*, “Pytorch: An imperative style, high-performance deep learning library,” in *Advances in neural information processing systems*, 2019, pp. 8026–8037.
[64] C. Vondrick, D. Patterson, and D. Ramanan, “Efficiently scaling up crowdsourced video annotation,” *International Journal of Computer Vision*, pp. 1–21, 10.1007/s11263-012-0564-1. [Online]. Available: http://dx.doi.org/10.1007/s11263-012-0564-1
[65] G. Bradski, “The OpenCV Library,” *Dr. Dobb’s Journal of Software Tools*, 2000.

Patrick Emami received his B.Sc. in Computer Engineering from the University of Florida in 2016. He is pursuing a Ph.D. degree in Computer Science at the University of Florida. His research is focused on developing neural algorithms for the analysis and generation of multi-object spatiotemporal data.

Lily Elefteriadou received the Ph.D. degree in transportation planning and engineering from NYU (Tandon School of Engineering), NY, USA, in 1994. She is currently a Professor with the University of Florida, where she directs the University of Florida Transportation Institute. She has served as the principal investigator for several federal and state projects, funded by the National Cooperative Highway Research Program (NCHRP), the National Science Foundation (NSF), the Federal Highway Administration, PennDOT, and FDOT. She has authored a textbook Introduction to Traffic Flow Theory. She serves on the Editorial Board for the Transportation Research: Part B. Her research interests include traffic operations, traffic flow theory, and simulation. She is the Past Chair of the Transportation Research Board’s Highway Capacity and Quality of Service Committee. She is also a Past Chair of the Executive Board of the Council of University Transportation Centers and was the President of the ARTBA Research and Education Council from 2014 to 2015.

Sanjay Ranka (S’87–M’88–SM’01–F’02) received the Ph.D. degree in computer science from the University of Minnesota, Minneapolis, USA, in 1988. He was the Chief Technology Officer with Paramark, where he developed real-time optimization software for optimizing marketing campaigns. He was one of the main architects of the Syracuse Fortran 90D/HPF compiler. He is currently a Professor with the University of Florida. He has coauthored two books: Elements of Neural Networks (MIT Press) and Hypercube Algorithms (Springer Verlag). His research interests include data mining, informatics and grid computing for data-intensive applications in high energy physics, bioterrorism, and biomedical computing. He was a past member of the Parallel Compiler Runtime Consortium and the Message Passing Initiative Standards Committee. He is a fellow of the AAAS, an Advisory Board Member of the IEEE Technical Committee on Parallel Processing, and a member of IFIP Committee on System Modeling and Optimization. He serves on the Editorial Board for the Journal of Parallel and Distributed Computing.

Algorithm 1 Multi-sensor MOT

```
1:  Define: An empty track array $\mathcal{T} = \{\}$, the temporal window size $w$, the set $\mathcal{H} = \{\}$ of hypothesized tracklets over $w$; the minimum allowed tracklet length $m$, radar distance threshold $\rho$, joint calibration matrix $\mathbf{P}$
2:  while still running do
3:      Gather $M$ tracklets from the last $w$ video frames $\mathcal{V} = \{v_i\}_{i=1}^{M}$ using MobileDR;
4:      for all $v \in \mathcal{V}$ do
5:          if length of $v_i > m$ then
6:              Project $v_i$ to world coordinates with $\mathbf{P}$;
7:          else
8:              $\mathcal{V} \leftarrow \mathcal{V} \setminus v_i$;
9:          end if
10:     end for
11:     Gather radar tracklets $\mathcal{R} = \{r_j\}_{j=1}^{N}$ from radar scans spanning $w$;
12:     for all $r \in \mathcal{R}$ do
13:         if length of $r_j \leq m$ or $r_j \leq \rho$ then
14:             $\mathcal{R} \leftarrow \mathcal{R} \setminus r_j$;
15:         end if
16:     end for
17:     Compute matches $\mathcal{M}$ and unmatched tracklets $\mathcal{U}$ with relaxed linear assignment on $(\mathcal{V}, \mathcal{R})$;
18:     $\mathcal{H} \leftarrow \mathcal{H} \cup \mathcal{U}$;
19:     for all matches $\in \mathcal{M}$ do
20:         Fuse the spatial and appearance information of match $(v_i, r_j)$ into tracklet $f$ using averaging;
21:         $\mathcal{H} \leftarrow \mathcal{H} \cup f$;
22:     end for
23:     Update track array $\mathcal{T}$ with hypothesis tracks $\mathcal{H}$ using relaxed linear assignment;
24: end while
```

Appendix

First we provide more details on the relaxed linear assignment algorithms used for cross-sensor tracklet matching and extending the temporal tracklet window. Then we discuss extra details about the experimental setup. See Algorithm 1 for pseudo-code of the multi-sensor framework.

A. Cross-sensor tracklet association

The association algorithm measures the relative similarity between N video and M radar tracklets with the Euclidean dynamic time warping (DTW) distance [47]. The DTW is an efficient way to measure the similarity of two 2D curves and is invariant to temporal stretching and shrinking (see Figure 8 for a visual illustration).

1) Algorithm setup: We define τ to be the maximum allowable DTW distance between any two pairs of tracklets and we define ρ to be a threshold for the minimum distance to the intersection for the centroid of a tracklet. We only fuse video and radar tracklets farther than ρ meters from the intersection; unmatched video tracklets farther than ρ are labeled as false positives, and all radar tracklets closer than ρ are labeled as false positives (due to their unreliability from the cosine effect).

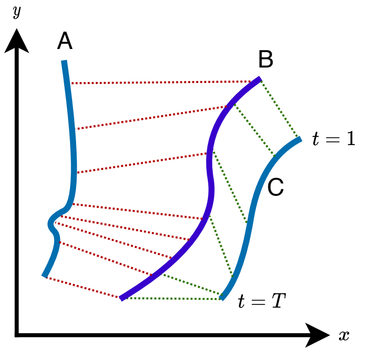


Figure 8: Euclidean DTW computation between video and radar tracklets. For one video tracklet (B) and two radar tracklets (A,C) of length T timestamps, the DTW computes the best possible spatiotemporal alignment between candidate matches using dynamic programming and outputs a score. The pair (B, A) has a higher DTW score than (B, C) because the shapes of the trajectories are harder to align. Hence, (B,C) is more likely to be selected.

We set τ to $3,000$ and ρ to 70 meters in our experiments by visualizing the tracker outputs on a held-out tuning set of manually collected multi-sensor data. We do not spend much effort optimizing these parameters as they are likely to be necessary to adjust depending on the installation location of the sensors and the geometry of the traffic intersection. Also, let $\mathcal{M}$ be an initially empty set of matched tracklets and $\mathcal{U}$ by an initially empty set of unmatched tracklets.

2) Algorithm steps:

1) Compute $\mathbf{D}$, the $N \times M$ DTW Euclidean distance matrix between video and radar tracklets
2) Ignore potential matches that do not overlap temporally by setting $\mathbf{D}[i,j] = \infty$
3) Augment $\mathbf{D}$ by adding an extra row and column with entries set to τ, so that if no match for the i^{th} video tracklet exists with DTW distance less than τ it gets assigned to a "dustbin" column
4) Using a fast linear program solver obtain the relaxed linear assignment with minimal cost, $\mathfrak{M}$, which is a list of length $\max(N, M)$ whose elements are indices from $\mathbf{D}$ specifying the matches
5) Unmatched radar tracklets and all video tracklets closer than ρ meters are added to $\mathcal{U}$
6) Valid matches where neither the video nor the radar tracklet has matched to the dustbin are added to $\mathcal{M}$

There is no constraint on how many tracklets can match to the dustbin row or column, whereas we enforce the one-to-one assignment constraint for all other rows and columns. As previously mentioned, unmatched video tracklets (farther than ρ meters away) are labeled as false positives.

B. Extending tracks from the previous time window

We maintain a set of tracks $\mathcal{T}$ which represents all objects believed to exist in the multi-sensor field of view up to but not including the current time window. The goal of the track extension algorithm is to efficiently update this set with $\mathcal{H} = \mathcal{M} \cup \mathcal{U}$. To accomplish this, we consider all potential pairings of A) the track states closest to the start of the current time window for each track $t \in \mathcal{T}$ with B) the temporally earliest tracklet state for all $h \in \mathcal{H}$. We use the inverse of the sensor calibration matrix $\mathbf{P}$ to project the world space x, y coordinates from the selected video and radar tracklet states into the image plane. Then, we compute a Euclidean distance matrix $\mathbf{I}$ based on the image plane coordinates. Computing this metric in the image plane makes it easy to capture the intuition that tracklet pairs for a small object (which appears small because is is far from the intersection) will have a small pixel distance. However, in world coordinates the distance between the same pair of tracklets may be arbitrarily large due to perspective distortion. We do not include velocity in our distance metric since vehicles that are turning have a large Euclidean distance between velocity vectors across time windows. All video tracks in $\mathcal{T}$ that do not have a corresponding radar track are automatically matched to the track in $\mathcal{H}$ with the same video track ID, if one exists, which we implement by setting the distance $\mathbf{I}[i, j] = 0$. This helps to keep the video MOT tracks intact and preserves the performance of the video MOT in the region closer than ρ meters from the intersection. Let $\mathfrak{M}$ be the list of matches from a relaxed linear assignment on $\mathbf{I}$ with dustbin threshold on the pixel distance p. We set this to 30 in our experiments, using a similar method to choose its value as with τ and ρ. Tracklets from $\mathcal{T}$ that have no match in $\mathcal{H}$ are assumed to have left the field of view and are removed from $\mathcal{T}$. Newly matched tracklets are assign the fused track ID of their match in $\mathcal{T}$. We update $\mathcal{T}$ with all newly unmatched and matched tracklets at the end.

C. More experiment details

1) Blob tracker: The blob tracker baseline for the Gainesville experiment is implemented by chaining together various OpenCV methods. First, we use the OpenCV Mixture-of-Gaussians background subtraction module to extract foreground masks from the input video. Then, we pass the foreground video to an open source implementation of a blob tracker (https://github.com/dghy/GUI_Blob_Tracker). This blob tracker extracts the local extrema from detected blobs in each video frame, processes them with a Kalman filter, and then uses the Hungarian algorithm to link the extrema over time into tracks. We found that background subtraction was a necessary preprocessing step since the blob tracker tended to pick up the road, signs, trees, etc.