Fast Vehicle Turning-Movement Counting using Localization-based Tracking

Derek Gloudemans and Daniel B. Work
Institute for Software Integrated Systems
Vanderbilt University
1025 16th Avenue S., Nashville, TN 37212
derek.gloudemans@vanderbilt.edu

Abstract

Despite the high utility of traffic volume and turning movement data, such data is still hard to come by for the vast majority of roadways and intersections in nearly every city. Edge computing devices offer a promising tool for recording turning movement data if lightweight algorithms can be designed to run in real-time with relatively modest computational complexity. To that end, this work presents Vehicle Turning-Movement Counting using Localization-based Tracking (LBT-Count). This method is fast because it never performs detection on a full frame. Instead, only a few portions of the image are cropped and used to detect objects within the frame. The method achieves competitive performance on the public evaluation server for Track 1 of the AI City Challenge (7th overall on the first 50% of data). Furthermore, we show that LBT-Count is 52% faster than an analogous counting algorithm utilizing a traditional tracking-by-detection framework on available challenge data.

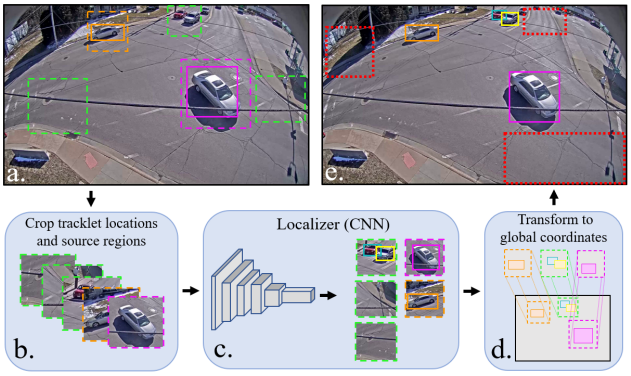

Figure 1. Proposed method (LBT-Count). **(a.)** Two types of regions within the current frame are of interest: manually-defined source regions (green dashed boxes) and expanded boxes (other dashed boxes) centered on tracked object *a priori* locations (solid boxes) which are predicted using information from previous frames. **(b.)** These regions are cropped from the overall frame. **(c.)** The localizer produces object bounding boxes within these crops. **(d.)** Bounding boxes are transformed back into global frame coordinates to update existing and initialize new tracklets. **(e.)** When tracked objects intersect sink regions (red dash), the associated vehicle movement is counted.

1. Introduction

Vehicle turning-movement counting is an essential tool for transportation planning. Accurate vehicle counts are necessary to determine roadway utilization, identify areas of congestion, and optimally allocate funding to maximally increase transportation quality of service within a constrained budget. Historically, these vehicle counts were performed manually using handheld electronic devices. Inductive-loop sensors [10] and radar sensors [27] offer automation of vehicle counting, but these infrastructure-based solutions are expensive, inflexible and only usable in the location in which they are installed, making counting of multiple vehicle movements at complex intersections difficult and costly. To combat these weaknesses, portable magnetic sensors have also been proposed [12].

Despite the relatively long and well-established usage of existing traffic-counting devices, the nexus of three key trends in the past 10 years make vehicle turning-movement counting a problem of renewed interest, as seen by this task's presence in the 2020 and 2021 AI City Challenges [38]. First, the development of fast and accurate image processing methods [28] allow for video data to be reliably and accurately used to provide information in transportation contexts. Second, state and federal transportation organizations have become increasingly interested in *intelligent transportation systems* (ITS) that both utilize and incorporate traffic data to allocate resources and even make transportation decisions in real-time. [45]. Lastly, *edge computing* devices continue to become cheaper, more computationally powerful, and more ubiquitous such that they now provide a feasible tool by which to study traffic in many cities [22, 24].

Vehicle turning-movement counting from video data is now a fairly well-studied problem [1, 4, 9, 13, 17, 18, 36, 39, 43, 44, 46, 47, 54, 56] and many accurate algorithms exist for this task. Nearly all such algorithms are based on the detect-track-count paradigm, where objects (vehicles) are first detected in a frame, then associated with detections from previous frames (tracked). Finally, vehicle trajectories are used to identify the turning movement of each vehicle through the camera field of view. Unfortunately, the accuracy of these methods cannot be realized in practice because slow object detection steps prevent these methods from processing incoming data in real-time. This is even more true when these algorithms are run edge devices, which are less computationally powerful than the server-grade computers on which algorithms are generally developed.

To address this issue, we leverage our recent work on a fast and accurate joint object detection and tracking method, *Localization-based Tracking* (LBT) [20]. This method avoids the slow detection step on most frames during tracking by using exisiting tracklet locations to provide a prior for object locations in the current frame. Regions corresponding to these tracklet locations are cropped and processed by a *localizer* (an object detector trained to locate a single object of interest within a crop) while other regions of the frame are ignored. Periodically, detection is performed on the overall frame to initialize new objects.

[20] The main contribution of our work is to introduce a new method for counting vehicle turning movements leveraging LBT to perform fast and accurate detection and tracking, which we call *Vehicle Turning-Movement Counting using Localization-based Tracking* (LBT-Count). New to this work, we utilize object source regions within a camera field of view and process these regions with the localizer at every frame such that detection is never performed on a full frame, yet new object tracklets can still be initialized at every frame. To the best of our knowledge, LBT-Count is the first traffic counting algorithm to explicitly avoid performing object detection on whole frames, and is one of only a few algorithms to break the detect-track-count paradigm for this task [39, 56].

An overview of the proposed approach is shown in Figure 1. At a frame n, (a). cropping boxes are generated based on i.) the predicted location of already-tracked objects, and ii.) predefined *source regions* for the camera field of view. The cropping boxes are (b.) cropped from the overall frame, and (c.) processed by the localizer to obtain predicted object bounding boxes. The bounding boxes are (d.) transformed into global frame coordinates, where bounding boxes from source region crops are used to initialize new objects, and boxes corresponding to object tracklets are used to update these tracklets. (e.) When tracked objects reach predefined *sink regions*, the combination of that object's source region of origin and the sink region are used to predict a turning movement for that object.

The rest of this article is organized as follows. In Section 2 we review existing approaches for vehicle turning-movement counting and related problems. Section 3 details our proposed method. Section 4 briefly describes Track 1 of the 2021 AI City Challenge on which we evaluate the proposed method, and Section 5 presents the competitive results of the evaluated method.

2. Related Work

Notable methods for tasks related to vehicle turning-movement counting are briefly reviewed, namely object detection, multiple object tracking and single-movement vehicle counting. Methods for vehicle turning-movement counting are then reviewed.

2.1. Object Detection

All vehicle counting methods rely on the accurate detection of vehicles within an image. Traditional approaches within the vehicle context relied on background subtraction and clustering [4, 5] or Gaussian mixture models [44, 51]. The vast majority of modern approaches leverage GPU-accelerated *convolutional neural networks* (CNNs) for fast and accurate object detection [14, 19, 29, 30, 33, 41, 42]. One accurate category of CNN architecture is the two-stage detector, in which a first stage extracts features and identifies regions likely to contain objects, and a second stage performs bounding box regression and classification on these proposals [42]. Faster one-stage object detectors skip the region proposal step to boost speed [33, 41]. Recent works have built upon this architecture, inspired by human object recognition behaviors to utilize object keypoints and keypoint-specific losses including corners [30], centers [14], or combinations of each at multiple scales [19]. Other approaches have explored ever-more-complex architectures for passing and aggregating information between layers. In [53] deeper aggregations and combinations of feature maps are used to create a more robust feature set, and in [29] a more nuanced neural architecture is learned for combining features at multiple scales, building on the powerful *feature pyramid network* (FPN) approach proposed in [33]. Most state-of-the-art detection methods cannot yet process images of modest size (e.g. 1920 x 1080 pixels) at the rate of modern video (30+ fps) [34].

2.2. Multiple Object Tracking

Multiple Object Tracking (MOT) is the task of associating objects in each frame temporally such that each unique object has the same label across all frames in which it appears. Some popular approaches utilize Euclidean distance between object tracklets and detections from the next frame [7, 15] or intersection-over-union-based comparison [8],

and utilize Kalman filters to predict tracklet locations for increased accuracy [7, 11, 15]. Other tracking algorithms incorporate visual features to associate objects across frames [32, 49, 50, 55].

Recent MOT methods have utilized information from the object tracking context to inform detection, performing detection and data association across frames jointly. One such framework is object re-detection, in which previous object locations are input to the detector as region proposals [6, 31] or heatmaps [57]. Other approaches pass pairs or larger sets of objects to the detector at each frame to both boost detection accuracy and aid in data association across frames [16, 40]. In [20], object locations from previous frames are used to crop relevant portions of the frame, and only these area are searched for objects, reducing CNN inference time.

2.3. Single-Movement Vehicle Counting

The task of single-movement vehicle counting or counting of vehicles passing a fixed line generally requires object detection, as well as object tracking to avoid double-counting the same vehicle in multiple frames. As noted in [21], single movement vehicle counting is still a challenging task in cases where camera field of view creates extremely high overlap between vehicles. In [5], an early algorithm for object counting is proposed utilizing background subtraction, blob fitting to cluster pixels into objects, and Kalman filtering to track vehicles across frames. [51] utilises Gaussian Mixture models for clustering background-subtracted pixels and compares each resulting cluster's convex hull area to its contained bright pixel area to explicitly predict object occlusion. [2, 3] use CNN-based object detectors and the Kanade-Lucas-Tomasi feature tracker to track and count objects. Similarly, [35, 37] utilize CNN object detectors and Kalman filtering for object tracking through the movement of interest, and [52] combines a cascade feature-based CNN with IOU tracking [8]. [36] also utilizes a weakly defined homography transformation into real-world coordinates to estimate each tracked vehicles length and inform vehicle classification. [48] makes use of foreground and background information to drastically reduce the feature space relative to image pixel-space before regressing object locations. [26] does not track objects, but instead maintains occupancy counts for several regions with the frame to logically determine when a vehicle should be counted.

2.4. Multiple Turning Movement Counting

The task of multiple turning movement counting is distinct from single-movement vehicle counting in that a movement uniquely defined by an object's origin and destination must be predicted for each counted object. As with single movement counting, multi-movement vehicle counting is still a challenging task especially when real-time performance is required. [18] utilizes a neural network to predict multiple vehicle turning movement counts at intersections given only aggregate approach traffic volume, which could effectively turn single-movement algorithms into multiple-movement counting algorithms, but this approach is not widely used. Nearly all algorithms for vehicle turning-movement follow the detect-track-count paradigm, where objects are detected in each frame, tracked across frames, and tracklets are subsequently categorized into turning movements, though [39] instead utilizes a joint tracking and detection method, Tracktor [6], and [56] instead directly regresses vehicle counts from an input video using a *Long-Short Term Memory* (LSTM) neural network to incorporate temporal information into the task. Most approaches for counting vehicle movements from trajectories utilize trajectory passage through unique sets of regions with the camera field of view to uniquely identify the relevant turning movement [4, 9, 13, 17, 39, 46], directly compare trajectories to canonical turning movements from each possible movement category [1, 36, 44, 54], or do some combination of the two [43, 47]. While many approaches in the first category require only a source and sink region to uniquely define a movement, some methods utilize larger sequences of regions to help distinguish between turning movements that occupy similar areas within a camera field of view [9, 46]. In the second category, the *longest common subset* (LCSS) shared by object trajectories and canonical turning movements is often used to assign turning movements to trajectories [44], but K-nearest neighbors clustering [1], scale-normalized trajectory similarity [54] and Hausdorff distance [36] are also used. [54] also scores each turning movement in terms of stability, completeness, and proximity to each object's trajectory, and smooths out anomalous points in each trajectory. [47] performs segmentation on trajectories and compares segments to known turning movement segments.

3. Methodology

This section describes *Vehicle Turning-Movement Counting using Localization-based Tracking* (LBT-Count) in detail.

The algorithmic process for an arbitrary frame n is shown in Figure 2. This method utilizes our previously-proposed Localization-based Tracking (LBT) framework [20], which leverages the tracking context to speed up object detection in a video. For each frame, regions of interest likely to contain objects (vehicles) are cropped from the overall frame. These crops are passed to a localizer (single object detector), and the resulting detected objects are used to update existing tracklets and initialize new tracklets. Whenever a tracked object enters a sink region of the frame, the path of the object is used to classify that object's turning movement, and thenceforth the vehicle is no longer tracked. We elaborate on each step of this process next.

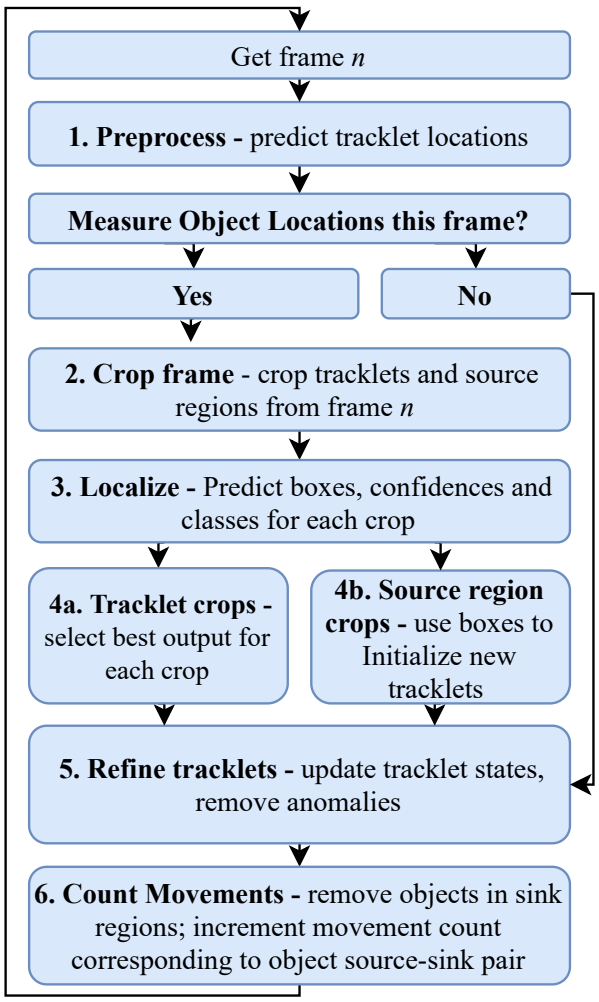

Figure 2. LBT-Count (proposed) process for a frame n.

Step 1: Preprocess Tracklet Locations

Throughout video processing, the state of each *tracklet* (tracked object) is estimated using a Kalman filter [25] as in Kalman filtering-based Intersection-over-Union tracking (KIOU) [8, 11]. A 2-dimensional, constant-velocity motion model is used for filtering each object's position. At each frame n, the filter is used to estimate *a priori* (estimated without measurement) locations of each existing tracklet.

In many video sequences, objects move predictably and can be tracked accurately without measuring their locations at every frame. In this work, a measurement is performed every f_{loc} frames (a parameter tuned per unique camera field of view). On all other frames, visual information is not used at all, and Step 2-4 are skipped.

Step 2: Crop Frame

Each *a priori* tracked object location (predicted using information from frames $\{0...n-1\}$ and expressed as a bounding box center x and y coordinate, width, and height) is expanded by a factor of b (to ensure that the tracked object is contained within the expanded box) and made square. The resulting boxes are used to crop the corresponding regions from the overall frame.

These crops only account for objects that were tracked for at least one previous frame. The original LBT framework initializes new objects for by performing detection on an overall frame periodically [20]. Instead, this work leverages the assumption that cameras are relatively static and new vehicles appear in a few, well-known regions within each camera field of view. This assumption is generally valid for traffic monitoring and for camera-equipped edge devices. We call the regions where new objects appear *source regions*. Source regions are manually identified once for each camera field of view. Figure 3 shows example source regions for a few camera fields of view. In addition to cropped regions based on existing object tracklets, each source region is also cropped to localize potential new vehicles. All crops are resized to square images of a standard size c_s pixels.

In each crop, regions containing visual information likely to mislead the localizer and reduce tracking accuracy are blacked out. These can include regions that are always misleading (e.g. parking lots and street-parked vehicles) and regions that are only misleading when initializing new objects (e.g. traffic on the opposing side of a highway). Regions of the former type are blacked out in all image crops, whereas regions of the latter type are blacked out only in crops corresponding to source regions such that existing objects can still be tracked through these regions. Figure 3 shows examples of each type of ignored region.

Step 3: Localize existing / Detect new objects

The task of locating a single object of interest within an image crop is called *localization*. Thus, we call the CNN-based object detector trained specifically for this task a *localizer*. A Retinanet with ResNet-50 FPN backbone trained specifically on crops of sized c_s is used as the localizer in this work [23, 33]. All crops from Step 2 are processed by the localizer, which outputs bounding boxes and corresponding confidences and class predictions for each crop.

Step 4a. Select Best Output for Tracklet Crops

The localizer outputs a set of bounding boxes for each crop. For crops generated from existing object tracklets, a single object is of interest. We parse the localizer outputs to select the best bounding box as in our previous work for LBT-extended KIOU [20]. Let i index the set of all tracked objects $\mathcal{O} := \{1, \cdots, i, \cdots, o_{max}\}$ and let j index the set of all localizer outputs $\mathcal{L} := \{1, \cdots, j, \cdots, l_{max}\}$. Each localizer output $\text{box}^l_{i,j}$ and corresponding confidence $\text{conf}_{i,j}$ is scored according to:

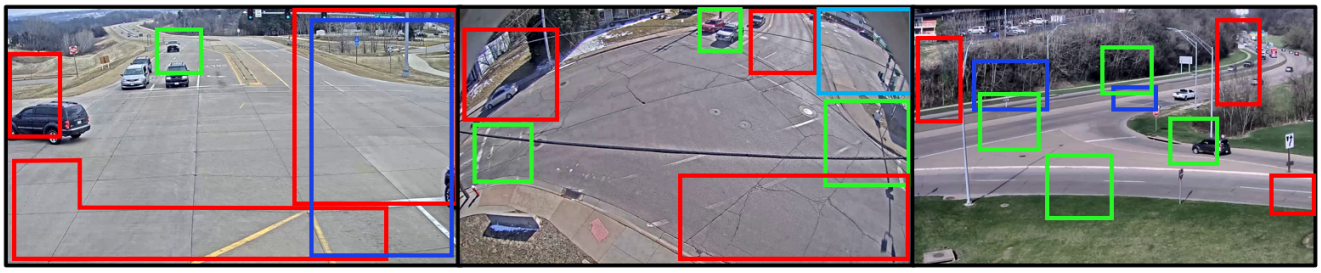

Figure 3. Examples of vehicle source regions (green), sink regions (red), regions that are blacked out in source crops only (dark blue) and regions that are blacked out in all crops (light blue) for several camera fields of view. Source regions are cropped and searched for new objects at each frame (Steps 2-3). Detected vehicles are tracked until they have travelled from a source region to a sink region, and the unique source-sink combination defines the vehicle's unique turning movement (Step 6).

$$\text{score}(\text{box}^l_{i,j}, \widetilde{\text{box}}_i) = W \times \text{conf}_{i,j} + \Phi(\text{box}^l_{i,j}, \widetilde{\text{box}}_i), \quad (1)$$

where $\widetilde{\text{box}}_i$ is the *a priori* object location for tracklet i, Φ is a function that computes the IOU similarity between two boxes, and W is a scalar parameter used to balance the two terms. The highest-scoring bounding box is selected as the localizer output for object i.

Step 4b. Initialize New Objects

The localizer outputs corresponding to source region crops are parsed differently than the outputs corresponding to tracklet crops. Instead, the outputs for each source are parsed using the following logic:

1. All output bounding boxes with confidence lower than σ_{min} (a tuned parameter) or with a predicted class other than {car, truck} are removed.
2. Non-maximal suppression is performed on all remaining bounding boxes.
3. All remaining bounding boxes with confidence lower than σ_{new} (a tuned parameter) are removed.
4. Any item that overlaps with an existing object by more than ϕ_{new} (a tuned parameter) in terms of intersection-over-union metric is removed.
5. All remaining bounding boxes are used to initialize new object tracklets, and the source region from which each object was initialized is recorded. The new object's speed is initialized in the Kalman filter as the estimated average speed of objects originating from the same source.

Note that although the localizer is trained primarily for single object detection, it is also capable of accurately detecting new objects within crops because the training process for the localizer is identical to the training process for a normal object detector except for the size of the images used for training.

Step 5. Refine Tracklets

The localizer-output bounding boxes from Step 4a are used to update the Kalman filter states for each existing tracklet. Then, the following steps are taken to remove anomalous tracklets.

- The localizer-output bounding box for each tracked object i has associated confidence conf_i and overlaps with that object's *a priori* location by ϕ_i (in terms of intersection-over-union metric). After initialization, each tracklet is required to have one localization where $\phi_i > \phi_{loc}$ and $\text{conf}_i > \sigma_{loc}$ within f_{max} frames of initialization, or else object tracklet i is no longer tracked. ϕ_{loc}, σ_{loc} and f_{max} are parameters tuned per camera.
- Objects are removed if they exceed reasonable bounding box size bounds $[s_{min}, s_{max}]$, defined per camera field of view.
- If two tracklet locations overlap by more than $\phi_{overlap}$ at frame n, the tracklet that has been tracked for fewer frames is pruned. Empirically, this most often occurs when two tracklets have been initialized for the same real object, though in a few cases this does result in occluded vehicles being pruned.
- Object tracklets that exit the frame are removed and no longer tracked.

Step 6. Count Movements

Just as objects tend to enter a frame at a few source regions, objects also exit the frame in a few, well-known *sink regions* within each camera field of view. These sink regions are manually labeled once for each camera field of view. After Step 5, each object is compared to each sink region. If the center of that object's bounding box falls within a sink region, that object is no longer tracked. Each unique source-sink combination identifies a vehicle turning movement of interest, so this corresponding vehicle movement is output by the algorithm. Example sink regions are shown for several camera fields of view in Figure 3.

At every frame for which object i is localized, the predicted class for that object is recorded. When an object reaches a sink region, the most frequently occurring class assignment for that object is output with that record. Empirically, the localizer has difficulty distinguishing between trucks and cars from some viewpoints, likely due to slightly differing definitions of which vehicles should be classified as trucks or cars across datasets. To boost classification accuracy, additional logic is used to classify trucks. When an object is initialized in a source region, if the object's starting bounding box size is more than s_{truck} times larger than the average object initialized at that source, n_{truck} additional predictions of truck are recorded to indicate a high likelihood that the object is a truck. For some camera fields of view in which objects are initialized far from the camera, initial bounding box size is not a good indicator of vehicle size. In these cases, objects that reach sink regions are compared against a baseline area a_{truck}, a parameter tuned per camera field of view. If the object bounding box area is greater than a_{truck}, the object is classified as a truck, and otherwise the object is classified as a car.

Logical limits are imposed on the frequency with which specific vehicle movements can occur. The minimum number of frames between movements f_{move} is set per vehicle movement, per camera view, and only source-sink combinations corresponding to valid vehicle movements are recorded. This helps to avoid double-counting vehicles in the event that multiple object tracklets correspond to a single real vehicle or that a tracklet from one source mistakenly begins tracking a vehicle from another source.

4. Experiments

The proposed method is evaluated on Track 1 of the 2021 AI City Challenge. Section 4.1 describes this challenge in more detail. Section 4.2 provides detailed parameter settings and implementation details for the evaluation.

4.1. AI City Challenge

Track 1 of the 2021 AI City Challenge requires multi-class, multi-movement vehicle counting on video sequences at intersections and along roadways. Thirty-one sequences from 20 distinct camera views are included, comprising about 9 hours of total video data all of which has resolution of at least 1280×960. Each camera field of view contains several vehicle movements of interest. To motivate the design of algorithms that can be evaluated in real-time on edge compute devices, the computational efficiency of vehicle counting algorithms are taken into account in addition to counting accuracy. Algorithms are assigned a score $S1$ according to the following formula:

$$S1 = 0.7 \times S1_{effectiveness} + 0.3 \times S1_{efficiency}$$

$S1_{effectiveness}$ uses cumulative vehicle counts at several times throughout each video sequence's overall length to evaluate counting effectiveness, weighting each time segment to help smooth jitters from vehicles counted near segment breakpoints. Cumulative count errors across all video sequences, turning movements and vehicle classes are normalized using the number of ground-truth vehicles within each cumulative count so that movements with more vehicles are counted more in the overall $S1_{effectiveness}$ score.

To partially account for the difference in operating speeds of various competitors' computing hardware, $S1_{efficiency}$ weights an algorithm's processing speed by the evaluating machine's speed at a set of benchmarking tasks relative to a baseline machine's speed on the same benchmark tasks. To compare these algorithms fairly in terms of speed, though, each algorithm must be run on the same compute hardware.

One half of the testing data is made available to challenge participants, with only a very small proportion of ground-truth labels provided such that supervised learning methods cannot feasibly be used. All submitted algorithms are evaluated on the full dataset, run on the same edge device (Nvidia Jetson NX development kit board.) As of submission, only aggregate $S1$ metrics from the first 50% of testing data are made public, so we report these scores in Section 5.

4.2. Parameter Settings and implementation Details

We use a Pytorch implementation of Retinanet with a ResNet50-FPN backbone for feature extraction. Because the scale of objects in image crops varies significantly from the scale of objects when detecting on whole frames, our localizer only is retrained for truck and car bounding box and class prediction per guidance from challenge organizers. Training makes no use of AI City Challenge data in any way. All code is run on a single GPU and 2 CPU cores (one of which is exclusively used for video decoding and frame buffering). Parameter settings or setting ranges for movement count tests are reported in Table 1.

Parameter	Value [Range]	Parameter	Value [Range]
σ_{min}	0.05	c_s	112
σ_{new}	[0.15 , 0.6]	s_{truck}	1.5
σ_{loc}	0.5	s_{min}	0
ϕ_{new}	0.3	s_{max}	1200
ϕ_{loc}	0.3	a_{truck}	160000
$\phi_{overlap}$	0.7	n_{truck}	100
f_{max}	[2 , 3]	W	0.5
f_{move}	[0 , 20]	b	[1.2,1.6]
f_{loc}	[2 , 5]		

Table 1. Parameter settings or ranges of settings for evaluation. Full parameter settings are available along with code for this work.

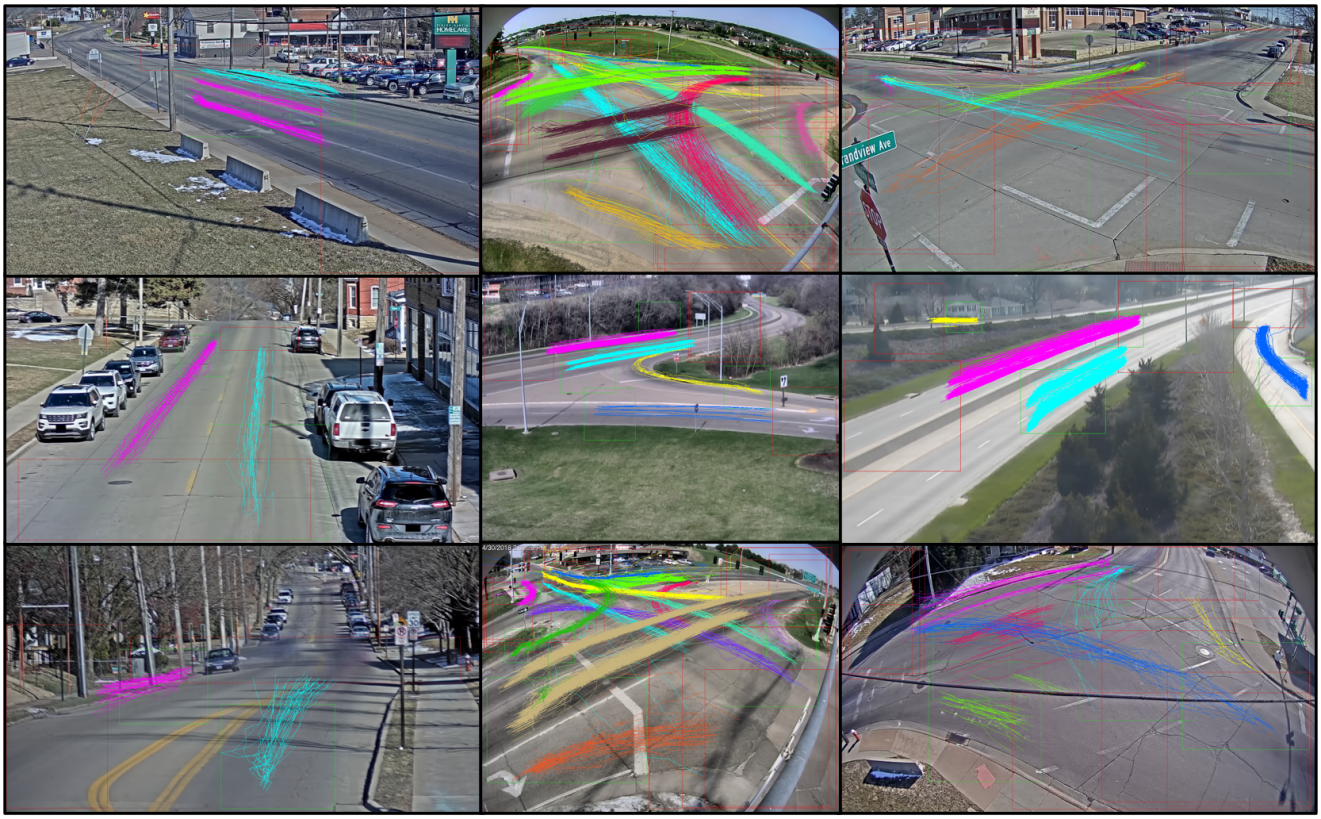

Figure 4. Vehicle paths for counted vehicle movements. Each movement is shown in a unique color per sequence. Faint green and red boxes denote source and sink regions, respectively.

5. Results

We report results on Track 1 of the 2021 AI City challenge. Section 5.1 reports overall score on the evaluation server, Section 5.2 presents qualitative results (as sufficient ground-truth data is purposely not made available for this challenge), and Section 5.3 provides a speed comparison of LBT-Count to an analogous tracking-by-detection-based counter following the detect-track-count framework.

5.1. Track 1 Leaderboard

Table 2 reports a comparison of all 17 algorithms submitted to the public Track 1 Challenge as part of the 2021 Nvidia AI City Challenge. Our algorithm (Team ID 95) places 7th in terms of $S1$ score on the 50% of testing data made publicly available, with $S1 = 0.8576$, $S1_{effectiveness} = 0.8549$ and $S1_{efficiency} = 0.8637$. LBT-Count processes the available videos at an average of 72.6 frames per second on a single GPU and 2 CPU cores.

5.2. Qualitative Results Analysis

Figure 4 shows the path of each object that was counted as a valid vehicle movement for several sequences, with object paths of each vehicle movement colored uniquely.

Team ID	Rank	S1 Score
37	1	0.9467
5	2	0.9459
8	3	0.9263
19	4	0.9249
118	5	0.9235
42	6	0.9157
95	**7**	**0.8576**
134	8	0.8449
153	9	0.8205
168	10	0.7545
144	11	0.7521
64	12	0.7506
86	13	0.6677
131	14	0.6548
133	15	0.4804
48	16	0.4205
77	17	0.3757

Table 2. $S1$ score for algorithms on 50% of testing data, evaluated on disparate machines.

A few observations are of note. First, there are very few anomalous paths indicating that few objects experi-

ence identity switches and generally tracking is quite accurate, even on sequences with many distinct turning movements. Second, object tracking ends as soon as objects reach sink regions; these sink regions were defined to maximize the chance of objects being captured correctly within these regions, and in some cases these regions are defined such that objects enter these sink regions before the exit the challenge-defined region of interest for a given camera field of view. We choose to emphasize accurate movement reporting over accurate movement time reporting. Based on these observations, the errors of our method likely fall predominantly into two categories: a.) false negatives when vehicles are lost somewhere within the region of interest or are never initially detected. b.) objects that are counted in the incorrect time bin due to premature tracklet termination at a sink box. Of these two sources of error, only a.) is of real concern for data quality as objects are at worst counted only a few seconds early or late.

5.3. Speed Comparison to Tracking by Detection

Lastly, to benchmark the impact of using Localization-based Tracking (LBT) rather than tracking-by-detection (TBD) for the object detection and tracking portions of our counting method, we implement a detect-track-count algorithm based on KIOU object tracking [8, 11]. We measure the speed of each method when a measurement step is performed at every frame. The same network structure is used for the localizer in LBT and the detector in TBD (Retinanet with ResNet50-FPN backbone). Table 3 reports the results.

LBT-Count is 52% faster than the detect-track-count (TBD) approach overall (20 fps vs 13.2 fps average). LBT is faster than TBD on 29 of 31 available test sequences, and achieves at least a 100% speedup on 19 of 31 sequences. The speedup of LBT is somewhat correlated to the number of crops (the sum of the number of tracked objects and the number of source regions for a camera field of view), as each cropped region requires additional computation to localize vehicles within it. Sequences with fewer than 19 crops per frame on average exclusively experience an increase in speed as a result of using the LBT framework.

Sequence	Speedup	Crops	LBT-Count fps	TBD fps
cam_14	**534%**	2.3	**30.9**	4.9
cam_16	**308%**	2.5	**40.8**	10.0
cam_17	**310%**	2.8	**40.5**	9.9
cam_20	**308%**	3.5	**39.3**	9.6
cam_19	**309%**	3.6	**38.3**	9.4
cam_18	**323%**	3.7	**39.1**	9.2
cam_13	**252%**	4.9	**37.7**	10.7
cam_15	**283%**	5.4	**36.8**	9.6
cam_1_dawn	**171%**	5.6	**42.0**	15.5
cam_12	**251%**	5.9	**36.5**	10.4
cam_2_rain	**97%**	6.0	**39.0**	19.8
cam_10	**251%**	6.4	**37.0**	10.5
cam_1_rain	**166%**	6.5	**40.7**	15.3
cam_1	**150%**	6.6	**38.2**	15.3
cam_2	**64%**	7.0	**29.7**	18.1
cam_3_rain	**94%**	7.8	**37.1**	19.1
cam_3	**42%**	7.9	**25.4**	17.9
cam_11	**249%**	9.2	**35.8**	10.3
cam_9	**223%**	9.4	**33.7**	10.4
cam_8	**231%**	12.0	**34.2**	10.3
cam_7_dawn	**131%**	15.6	**33.9**	14.7
cam_4_rain	**102%**	16.5	**30.7**	15.2
cam_6_snow	**158%**	18.6	**32.9**	12.7
cam_4_dawn	**68%**	19.1	**25.7**	15.3
cam_4	-13%	19.5	12.7	**14.6**
cam_6	**35%**	21.5	**18.3**	13.6
cam_5_dawn	**78%**	21.5	**24.8**	13.9
cam_7_rain	**89%**	21.6	**26.1**	13.9
cam_5_rain	**81%**	24.6	**25.2**	13.9
cam_7	**9%**	27.3	**12.0**	11.1
cam_5	-12%	27.9	11.9	**13.5**
Average	**52%**	11.4	**20.0**	13.2

Table 3. Speedup from using LBT-Count versus a tracking-by-detection-based counter (TBD). "Crops" indicates the average number of cropped regions processed by the localizer per frame in LBT-Count.

6. Conclusion

In this work, we present LBT-Count, a novel method for generating vehicle turning movement counts from raw video that breaks from the detect-track-count paradigm by localizing objects in smaller crops without ever performing detection on a full frame. We evaluate our method as part of Track 1 of the 2021 AI City Challenge and achieve competitive performance, and further show that the method significantly increases the speed of the detection and tracking portion of this task (52% relative to an analogous tracking-by-detection approach). In future work, a detailed investigation of the role of hyperparameter settings on performance is warranted; such an analysis requires ground truth data on vehicle turning movements, which was neither provided nor allowed for this challenge. Code for this method is available at https://github.com/DerekGloudemans/LBT-count/.

Acknowledgments

This material is based upon work supported by the National Science Foundation under Grant No. CNS-1837652 (Work) and by the National Science Foundation Graduate Research Fellowship under Grant No. DGE-1937963. This material is based upon work supported by the U.S. Department of Energy's Office of Energy Efficiency and Renewable Energy (EERE) award number CID DE-EE0008872. The views expressed herein do not necessarily represent the views of the U.S. Department of Energy or the United States Government.

References

[1] Awad Abdelhalim and Montasir Abbas. Towards real-time traffic movement count and trajectory reconstruction using virtual traffic lanes. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops*, pages 592–593, 2020.

[2] Mohamed A Abdelwahab. Accurate vehicle counting approach based on deep neural networks. In *2019 International Conference on Innovative Trends in Computer Engineering (ITCE)*, pages 1–5. IEEE, 2019.

[3] Z. Al-Ariny, M. A. Abdelwahab, M. Fakhry, and E. Hasaneen. An efficient vehicle counting method using mask r-cnn. In *2020 International Conference on Innovative Trends in Communication and Computer Engineering (ITCE)*, pages 232–237, 2020.

[4] Pablo Barcellos, Christiano Bouvié, Fabiano Lopes Escouto, and Jacob Scharcanski. A novel video based system for detecting and counting vehicles at user-defined virtual loops. *Expert Systems with Applications*, 42(4):1845–1856, 2015.

[5] Erhan Bas, A Murat Tekalp, and F Sibel Salman. Automatic vehicle counting from video for traffic flow analysis. In *2007 IEEE intelligent vehicles symposium*, pages 392–397. Ieee, 2007.

[6] Philipp Bergmann, Tim Meinhardt, and Laura Leal-Taixe. Tracking without bells and whistles. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 941–951, 2019.

[7] Alex Bewley, Zongyuan Ge, Lionel Ott, Fabio Ramos, and Ben Upcroft. Simple online and realtime tracking. In *2016 IEEE International Conference on Image Processing (ICIP)*, pages 3464–3468. IEEE, 2016.

[8] Erik Bochinski, Volker Eiselein, and Thomas Sikora. High-speed tracking-by-detection without using image information. In *2017 14th IEEE International Conference on Advanced Video and Signal Based Surveillance (AVSS)*, pages 1–6. IEEE, 2017.

[9] Nam Bui, Hongsuk Yi, and Jiho Cho. A vehicle counts by class framework using distinguished regions tracking at multiple intersections. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops*, pages 578–579, 2020.

[10] Chao Chen, Karl Petty, Alexander Skabardonis, Pravin Varaiya, and Zhanfeng Jia. Freeway performance measurement system: mining loop detector data. *Transportation Research Record*, 1748(1):96–102, 2001.

[11] S. Chen and C. Shao. Python implementation of the kalman-iou tracker. `https://github.com/siyuanc2/kiout`. Accessed: 2021-03-12.

[12] Thou-Ho Chen, Yu-Feng Lin, and Tsong-Yi Chen. Intelligent vehicle counting method based on blob analysis in traffic surveillance. In *Second International Conference on Innovative Computing, Informatio and Control (ICICIC 2007)*, pages 238–238. IEEE, 2007.

[13] Zhe Dai, Huansheng Song, Xuan Wang, Yong Fang, Xu Yun, Zhaoyang Zhang, and Huaiyu Li. Video-based vehicle counting framework. *IEEE Access*, 7:64460–64470, 2019.

[14] Kaiwen Duan, Song Bai, Lingxi Xie, Honggang Qi, Qingming Huang, and Qi Tian. Centernet: Keypoint triplets for object detection. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 6569–6578, 2019.

[15] Jianxin Fang, Huadong Meng, Hao Zhang, and Xiqin Wang. A low-cost vehicle detection and classification system based on unmodulated continuous-wave radar. In *2007 IEEE Intelligent Transportation Systems Conference*, pages 715–720. IEEE, 2007.

[16] Christoph Feichtenhofer, Axel Pinz, and Andrew Zisserman. Detect to track and track to detect. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 3038–3046, 2017.

[17] Jan Folenta, Jakub Spanhel, Vojtech Bartl, and Adam Herout. Determining vehicle turn counts at multiple intersections by separated vehicle classes using cnns. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops*, pages 596–597, 2020.

[18] Mohammad Shareef Ghanim and Khaled Shaaban. Estimating turning movements at signalized intersections using artificial neural networks. *IEEE Transactions on Intelligent Transportation Systems*, 20(5):1828–1836, 2018.

[19] Golnaz Ghiasi, Tsung-Yi Lin, and Quoc V. Le. Nas-fpn: Learning scalable feature pyramid architecture for object detection. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2019.

[20] Derek Gloudemans and Daniel B. Work. Localization-based tracking. *arXiv preprint arXiv:2104.05823*, 2021.

[21] Ricardo Guerrero-Gómez-Olmedo, Beatriz Torre-Jiménez, Roberto López-Sastre, Saturnino Maldonado-Bascón, and Daniel Onoro-Rubio. Extremely overlapping vehicle counting. In *Iberian Conference on Pattern Recognition and Image Analysis*, pages 423–431. Springer, 2015.

[22] Gerhard P Hancke, Gerhard P Hancke Jr, et al. The role of advanced sensing in smart cities. *Sensors*, 13(1):393–425, 2013.

[23] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016.

[24] Ling Hu and Qiang Ni. Iot-driven automated object detection algorithm for urban surveillance systems in smart cities. *IEEE Internet of Things Journal*, 5(2):747–754, 2017.

[25] Rudolph Emil Kalman. A new approach to linear filtering and prediction problems. *Transactions of the ASME–Journal of Basic Engineering*, 82(Series D):35–45, 1960.

[26] Shiva Kamkar and Reza Safabakhsh. Vehicle detection, counting and classification in various conditions. *IET Intelligent Transport Systems*, 10(6):406–413, 2016.

[27] Lawrence A Klein, Milton K Mills, David RP Gibson, et al. Traffic detector handbook: Volume i. Technical report, Turner-Fairbank Highway Research Center, 2006.

[28] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. *Advances in neural information processing systems*, 25:1097–1105, 2012.

[29] Shiyi Lan, Zhou Ren, Yi Wu, Larry S. Davis, and Gang Hua. Saccadenet: A fast and accurate object detector. In *Proceedings of the IEEE/CVF Conference on Computer Vision and*

Pattern Recognition (CVPR), June 2020.

[30] Hei Law and Jia Deng. Cornernet: Detecting objects as paired keypoints. In *Proceedings of the European conference on computer vision (ECCV)*, pages 734–750, 2018.

[31] Wei Li, Yuanjun Xiong, Shuo Yang, Siqi Deng, and Wei Xia. Smot: Single-shot multi object tracking. *arXiv preprint arXiv:2010.16031*, 2020.

[32] Chao Liang, Zhipeng Zhang, Yi Lu, Xue Zhou, Bing Li, Xiyong Ye, and Jianxiao Zou. Rethinking the competition between detection and reid in multi-object tracking. *arXiv preprint arXiv:2010.12138*, 2020.

[33] Tsung-Yi Lin, Priya Goyal, Ross Girshick, Kaiming He, and Piotr Dollar. Focal loss for dense object detection. In *The IEEE International Conference on Computer Vision (ICCV)*, Oct 2017.

[34] Tsung-Yi Lin, Michael Maire, Serge Belongie, James Hays, Pietro Perona, Deva Ramanan, Piotr Dollár, and C Lawrence Zitnick. Microsoft coco: Common objects in context. In *European conference on computer vision*, pages 740–755. Springer, 2014.

[35] Chenghuan Liu, Du Q Huynh, Yuchao Sun, Mark Reynolds, and Steve Atkinson. A vision-based pipeline for vehicle counting, speed estimation, and classification. *IEEE Transactions on Intelligent Transportation Systems*, 2020.

[36] Zhongji Liu, Wei Zhang, Xu Gao, Hao Meng, Xiao Tan, Xiaoxing Zhu, Zhan Xue, Xiaoqing Ye, Hongwu Zhang, Shilei Wen, et al. Robust movement-specific vehicle counting at crowded intersections. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops*, pages 614–615, 2020.

[37] Lu Lou, Qi Zhang, Chunfang Liu, Minlan Sheng, Jun Liu, and Huimin Song. Detecting and counting the moving vehicles using mask r-cnn. In *2019 IEEE 8th Data Driven Control and Learning Systems Conference (DDCLS)*, pages 987–992. IEEE, 2019.

[38] Milind Naphade, Shuo Wang, David C. Anastasiu, Zheng Tang, Ming-Ching Chang, Xiaodong Yang, Liang Zheng, Anuj Sharma, Rama Chellappa, and Pranamesh Chakraborty. The 4th ai city challenge. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR) Workshops*, page 2665–2674, June 2020.

[39] Andres Ospina and Felipe Torres. Countor: Count without bells and whistles. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops*, pages 600–601, 2020.

[40] Bo Pang, Yizhuo Li, Yifan Zhang, Muchen Li, and Cewu 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*, pages 6308–6318, 2020.

[41] Joseph Redmon and Ali Farhadi. Yolov3: An incremental improvement. *arXiv preprint arXiv:1804.02767*, 2018.

[42] Shaoqing Ren, Kaiming He, Ross Girshick, and Jian Sun. Faster r-cnn: Towards real-time object detection with region proposal networks. In *Advances in neural information processing systems*, pages 91–99, 2015.

[43] Mohammad Shokrolah Shirazi and Brendan Morris. A typical video-based framework for counting, behavior and safety analysis at intersections. In *2015 IEEE Intelligent Vehicles Symposium (IV)*, pages 1264–1269. IEEE, 2015.

[44] Mohammad Shokrolah Shirazi and Brendan Tran Morris. Trajectory prediction of vehicles turning at intersections using deep neural networks. *Machine Vision and Applications*, 30(6):1097–1109, 2019.

[45] Katherine F Turnbull. Critical issues in transportation. *TR NEWS*, 2019.

[46] Wei Wang, Tim Gee, Jeff Price, and Hairong Qi. Real time multi-vehicle tracking and counting at intersections from a fisheye camera. In *2015 IEEE Winter Conference on Applications of Computer Vision*, pages 17–24. IEEE, 2015.

[47] Zhihui Wang, Bing Bai, Yujun Xie, Tengfei Xing, Bineng Zhong, Qinqin Zhou, Yiping Meng, Bin Xu, Zhichao Song, Pengfei Xu, et al. Robust and fast vehicle turncounts at intersections via an integrated solution from detection, tracking and trajectory modeling. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops*, pages 610–611, 2020.

[48] Zilei Wang, Xu Liu, Jiashi Feng, Jian Yang, and Hongsheng Xi. Compressed-domain highway vehicle counting by spatial and temporal regression. *IEEE Transactions on Circuits and Systems for Video Technology*, 29(1):263–274, 2017.

[49] Zhongdao Wang, Liang Zheng, Yixuan Liu, Yali Li, and Shengjin Wang. Towards real-time multi-object tracking. *arXiv preprint arXiv:1909.12605*, 2019.

[50] Nicolai Wojke, Alex Bewley, and Dietrich Paulus. Simple online and realtime tracking with a deep association metric. In *2017 IEEE international conference on image processing (ICIP)*, pages 3645–3649. IEEE, 2017.

[51] Yingjie Xia, Xingmin Shi, Guanghua Song, Qiaolei Geng, and Yuncai Liu. Towards improving quality of video-based vehicle counting method for traffic flow estimation. *Signal Processing*, 120:672–681, 2016.

[52] Yomna Youssef and Mohamed Elshenawy. Automatic vehicle counting and tracking in aerial video feeds using cascade region-based convolutional neural networks and feature pyramid networks. *Transportation Research Record*, page 0361198121997833, 2021.

[53] Fisher Yu, Dequan Wang, Evan Shelhamer, and Trevor Darrell. Deep layer aggregation. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 2403–2412, 2018.

[54] Lijun Yu, Qianyu Feng, Yijun Qian, Wenhe Liu, and Alexander G Hauptmann. Zero-virus: Zero-shot vehicle route understanding system for intelligent transportation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops*, pages 594–595, 2020.

[55] Yifu Zhan, Chunyu Wang, Xinggang Wang, Wenjun Zeng, and Wenyu Liu. A simple baseline for multi-object tracking. *arXiv preprint arXiv:2004.01888*, 2020.

[56] Shanghang Zhang, Guanhang Wu, Joao P. Costeira, and Jose M. F. Moura. Fcn-rlstm: Deep spatio-temporal neural networks for vehicle counting in city cameras. In *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*, Oct 2017.

[57] Xingyi Zhou, Vladlen Koltun, and Philipp Krähenbühl. Tracking objects as points. In *European Conference on Computer Vision*, pages 474–490. Springer, 2020.