Tracking Objects as Points

Xingyi Zhou[1], Vladlen Koltun[2], and Philipp Krähenbühl[1]

[1]UT Austin, [2]Intel Labs

Abstract. Tracking has traditionally been the art of following interest points through space and time. This changed with the rise of powerful deep networks. Nowadays, tracking is dominated by pipelines that perform object detection followed by temporal association, also known as tracking-by-detection. We present a simultaneous detection and tracking algorithm that is simpler, faster, and more accurate than the state of the art. Our tracker, CenterTrack, applies a detection model to a pair of images and detections from the prior frame. Given this minimal input, CenterTrack localizes objects and predicts their associations with the previous frame. That's it. CenterTrack is simple, online (no peeking into the future), and real-time. It achieves 67.8% MOTA on the MOT17 challenge at 22 FPS and 89.4% MOTA on the KITTI tracking benchmark at 15 FPS, setting a new state of the art on both datasets. CenterTrack is easily extended to monocular 3D tracking by regressing additional 3D attributes. Using monocular video input, it achieves 28.3% AMOTA@0.2 on the newly released nuScenes 3D tracking benchmark, substantially outperforming the monocular baseline on this benchmark while running at 28 FPS.

Keywords: Multi-object tracking; Conditioned detection; 3D object tracking.

1 Introduction

In early computer vision, tracking was commonly phrased as following interest points through space and time [36, 43]. Early trackers were simple, fast, and reasonably robust. However, they were liable to fail in the absence of strong low-level cues such as corners and intensity peaks. With the advent of high-performing object detection models [9, 31], a powerful alternative emerged: tracking-by-detection (or more precisely, tracking-after-detection) [2, 41, 50]. These models rely on a given accurate recognition to identify objects and then link them up through time in a separate stage. Tracking-by-detection leverages the power of deep-learning-based object detectors and is currently the dominant tracking paradigm. Yet the best-performing object trackers are not without drawbacks. Many rely on slow and complex association strategies to link detected boxes through time [14, 41, 48, 50]. Recent work on *simultaneous detection and tracking* [1, 8] has made progress in alleviating some of this complexity. Here, we show how combining ideas from point-based tracking and simultaneous detection and tracking further simplifies tracking.

We present a point-based framework for joint detection and tracking, referred to as CenterTrack. Each object is represented by a single point at the center of its bounding box. This center point is then tracked through time (Figure 1). Specifically, we adopt the

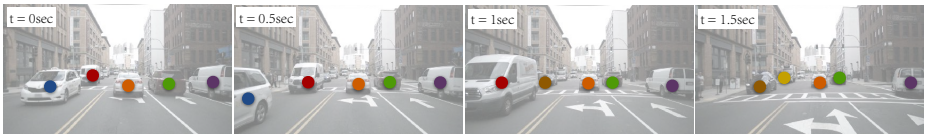

Fig. 1: We track objects by tracking their centers. We learn a 2D offset between two adjacent frames and associate them based on center distance.

recent CenterNet detector to localize object centers [56]. We condition the detector on two consecutive frames, as well as a heatmap of prior tracklets, represented as points. We train the detector to also output an offset vector from the current object center to its center in the previous frame. We learn this offset as an attribute of the center point at little additional computational cost. A greedy matching, based solely on the distance between this predicted offset and the detected center point in the previous frame, suffices for object association. The tracker is end-to-end trainable and differentiable.

Tracking objects as points simplifies two key components of the tracking pipeline. First, it simplifies tracking-conditioned detection. If each object in past frames is represented by a single point, a constellation of objects can be represented by a heatmap of points [4]. Our tracking-conditioned detector directly ingests this heatmap and reasons about all objects jointly when associating them across frames. Second, point-based tracking simplifies object association across time. A simple displacement prediction, akin to sparse optical flow, allows objects in different frames to be linked. This displacement prediction is conditioned on prior detections. It learns to jointly detect objects in the current frame and associate them to prior detections.

While the overall idea is simple, subtle details matter in making this work. Tracked objects in consecutive frames are highly correlated. With the previous-frame heatmap given as input, CenterTrack could easily learn to repeat the predictions from the preceding frame, and thus refuse to track without incurring a large training error. We prevent this through an aggressive data-augmentation scheme during training. In fact, our data augmentation is aggressive enough for the model to learn to track objects from *static images*. That is, CenterTrack can be successfully trained on static image datasets (with "hallucinated" motion), with no real video input.

CenterTrack is purely local. It only associates objects in adjacent frames, without reinitializing lost long-range tracks. It trades the ability to reconnect long-range tracks for simplicity, speed, and high accuracy in the local regime. Our experiments indicate that this trade-off is well worth it. CenterTrack outperforms complex tracking-by-detection strategies on the MOT [28] and KITTI [12] tracking benchmarks. We further apply the approach to monocular 3D object tracking on the nuScenes dataset [3]. Our monocular tracker achieves 28.3% AMOTA@0.2, outperforming the monocular baseline by a factor of 3, while running at 22 FPS. It can be trained on labelled video sequences, if available, or on static images with data augmentation. Code is available at https://github.com/xingyizhou/CenterTrack.

2 Related work

Tracking-by-detection. Most modern trackers [2, 7, 23, 33, 35, 41, 47, 50, 58] follow the tracking-by-detection paradigm. An off-the-shelf object detector [9, 30, 31, 51] first finds all objects in each individual frame. Tracking is then a problem of bounding box association. SORT [2] tracks bounding boxes using a Kalman filter and associates each bounding box with its highest overlapping detection in the current frame using bipartite matching. DeepSORT [47] augments the overlap-based association cost in SORT with appearance features from a deep network. More recent approaches focus on increasing the robustness of object association. Tang et al. [41] leverage person-reidentification features and human pose features. Xu et al. [50] take advantage of the spatial locations over time. BeyondPixel [35] uses additional 3D shape information to track vehicles.

These methods have two drawbacks. First, the data association discards image appearance features [2] or requires a computationally expensive feature extractor [10, 35, 41, 50]. Second, detection is separated from tracking. In our approach, association is almost free. Association is *learned* jointly with detection. Also, our detector takes the previous tracking results as an input, and can learn to recover missing or occluded objects from this additional cue.

Joint detection and tracking. A recent trend in multi-object tracking is to convert existing detectors into trackers and combine both tasks in the same framework. Feichtenhofer et al. [8] use a siamese network with the current and past frame as input and predict inter-frame offsets between bounding boxes. Integrated detection [55] uses tracked bounding boxes as additional region proposals to enhance detection, followed by bipartite-matching-based bounding-box association. Tracktor [1] removes the box association by directly propagating identities of region proposals using bounding box regression. In video object detection, Kang et al. [16, 17] feed stacked consecutive frames into the network and do detection for a whole video segment. And Zhu et al. [59] use flow to warp intermediate features from previous frames to accelerate inference.

Our method belongs to this category. The difference is that all of these works adopt the FasterRCNN framework [31], where the tracked boxes are used as region proposals. This assumes that bounding boxes have a large overlap between frames, which is not true in low-framerate regimes. As a consequence, Tracktor [1] requires a motion model [5, 6] for low-framerate sequences. Our approach instead provides the tracked predictions as an additional point-based heatmap input to the network. The network is then able to reason about and match objects anywhere in its receptive field even if the boxes have no overlap at all.

Motion prediction. Motion prediction is another important component in a tracking system. Early approaches [2, 47] used Kalman filters to model object velocities. Held et al. [13] use a regression network to predict four scalars for bounding box offset between frames for single-object tracking. Xiao et al. [49] utilize an optical flow estimation network to update joint locations in human pose tracking. Voigtlaender et al. [45] learn a high-dimensional embedding vector for object identities for simultaneous object tracking and segmentation. Our center offset is analogous to sparse optical flow, but is learned together with the detection network and does not require dense supervision.

Heatmap-conditioned keypoint estimation. Feeding the model predictions as an additional input to a model works across a wide range of vision tasks [44], especially

for keypoint estimation [4, 11, 29]. Auto-context [44] feeds the mask prediction back into the network. Iterative-Error-Feedback (IEF) [4] takes another step by rendering predicted keypoint coordinates into heatmaps. PoseFix [29] generates heatmaps that simulate test errors for human pose refinement.

Our tracking-conditioned detection framework is inspired by these works. A rendered heatmap of prior keypoints [4, 11, 29, 44] is especially appealing in tracking for two reasons. First, the information in the previous frame is freely available and does not slow down the detector. Second, conditional tracking can reason about occluded objects that may no longer be visible in the current frame. The tracker can simply learn to keep those detections from the prior frame around.

3D object detection and tracking. 3D trackers replace the object detection component in standard tracking systems with 3D detection from monocular images [30] or 3D point clouds [37, 57]. Tracking then uses an off-the-shelf identity association model. For example, 3DT [14] detects 2D bounding boxes, estimates 3D motion, and uses depth and order cues for matching. AB3D [46] achieves state-of-the-art performance by combining a Kalman filter with accurate 3D detections [37].

3 Preliminaries

Our method, CenterTrack, builds on the CenterNet detector [56]. CenterNet takes a single image $I \in \mathbb{R}^{W \times H \times 3}$ as input and produces a set of detections $\{(\mathbf{p}_i, \mathbf{s}_i)\}_{i=0}^{N-1}$ for each class $c \in \{0, \dots, C-1\}$. CenterNet identifies each object through its center point $\mathbf{p} \in \mathbb{R}^2$ and then regresses to a height and width $\mathbf{s} \in \mathbb{R}^2$ of the object's bounding box. Specifically, it produces a low-resolution heatmap $\hat{Y} \in [0,1]^{\frac{W}{R} \times \frac{H}{R} \times C}$ and a size map $\hat{S} \in \mathbb{R}^{\frac{W}{R} \times \frac{H}{R} \times 2}$ with a downsampling factor $R = 4$. Each local maximum $\hat{\mathbf{p}} \in \mathbb{R}^2$ (also called peak, whose response is the strongest in a 3×3 neighborhood) in the heatmap $\hat{Y}$ corresponds to a center of a detected object with confidence $\hat{w} = \hat{Y}_{\hat{\mathbf{p}}}$ and object size $\hat{\mathbf{s}} = \hat{S}_{\hat{\mathbf{p}}}$.

Given an image with a set of annotated objects $\{\mathbf{p}_0, \mathbf{p}_1, \dots\}$, CenterNet uses a training objective based on the focal loss [22, 25]:

$$L_k = \frac{1}{N} \sum_{xyc} \begin{cases} (1 - \hat{Y}_{xyc})^{\alpha} \log(\hat{Y}_{xyc}) & \text{if } Y_{xyc} = 1 \\ (1 - Y_{xyc})^{\beta} (\hat{Y}_{xyc})^{\alpha} \log(1 - \hat{Y}_{xyc}) & \text{otherwise} \end{cases}, \tag{1}$$

where $Y \in [0,1]^{\frac{W}{R} \times \frac{H}{R} \times C}$ is a ground-truth heatmap corresponding to the annotated objects. N is the number of objects, and $\alpha = 2$ and $\beta = 4$ are hyperparameters of the focal loss. For each center $\mathbf{p}$ of class c, we render a Gaussian-shaped peak into $Y_{:,:,c}$ using a rendering function $Y = \mathcal{R}(\{\mathbf{p}_0, \mathbf{p}_1, \dots\})$ [22]. Formally, the rendering function at position $\mathbf{q} \in \mathbb{R}^2$ is defined as

$$\mathcal{R}_{\mathbf{q}}(\{\mathbf{p}_0, \mathbf{p}_1, \dots\}) = \max_i \exp\left(-\frac{(\mathbf{p}_i - \mathbf{q})^2}{2\sigma_i^2}\right).$$

The Gaussian kernel σ_i is a function of the object size [22].

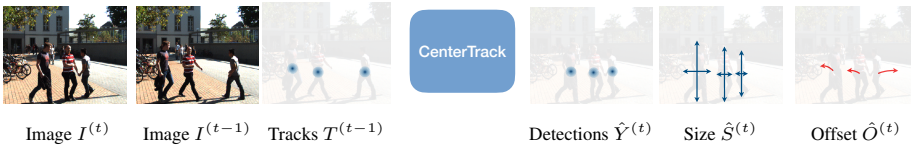

Fig. 2: Illustration of our framework. The network takes the current frame, the previous frame, and a heatmap rendered from tracked object centers as inputs, and produces a center detection heatmap for the current frame, the bounding box size map, and an offset map. At test time, object sizes and offsets are extracted from peaks in the heatmap.

The size prediction is only supervised at the center locations. Let $\mathbf{s}_i$ be the bounding box size of the i-th object at location $\mathbf{p}_i$. Size prediction is learned by regression

$$L_{size} = \frac{1}{N}\sum_{i=1}^{N} |\hat{S}_{\mathbf{p}_i} - \mathbf{s}_i|. \tag{2}$$

CenterNet further regresses to a refined center local location using an analogous L1 loss L_{loc}. The overall loss of CenterNet is a weighted sum of all three loss terms: focal loss, size, and local location regression.

4 Tracking objects as points

We approach tracking from a local perspective. When an object leaves the frame or is occluded and reappears, it is assigned a new identity. We thus treat tracking as the problem of propagating detection identities across *consecutive* frames, without re-establishing associations across temporal gaps.

At time t, we are given an image of the current frame $I^{(t)} \in \mathbb{R}^{W\times H\times 3}$ and the previous frame $I^{(t-1)} \in \mathbb{R}^{W\times H\times 3}$, as well as the tracked objects in the previous frame $T^{(t-1)} = \{b_0^{(t-1)}, b_1^{(t-1)}, \ldots\}_i$. Each object $b = (\mathbf{p}, \mathbf{s}, w, id)$ is described by its center location $\mathbf{p} \in \mathbb{R}^2$, size $\mathbf{s} \in \mathbb{R}^2$, detection confidence $w \in [0, 1]$, and unique identity $id \in \mathbb{I}$. Our aim is to detect and track objects $T^{(t)} = \{b_0^{(t)}, b_1^{(t)}, \ldots\}$ in the current frame t, and assign objects that appear in both frames a consistent id.

There are two main challenges here. The first is finding all objects in every frame – including occluded ones. The second challenge is associating these objects through time. We address both via a single deep network, trained end-to-end. Section 4.1 describes a tracking-conditioned detector that leverages tracked detections from the previous frame to improve detection in the current frame. Section 4.2 then presents a simple offset prediction scheme that is able to link detections through time. Finally, Sections 4.3 and 4.4 show how to train this detector from video or static image data.

4.1 Tracking-conditioned detection

As an object detector, CenterNet already infers most of the required information for tracking: object locations $\hat{\mathbf{p}}$, their size $\hat{\mathbf{s}} = \hat{S}_{\hat{\mathbf{p}}}$, and a confidence measure $\hat{w} = \hat{Y}_{\hat{\mathbf{p}}}$.

However, it is unable to find objects that are not directly visible in the current frame, and the detected objects may not be temporally coherent. One natural way to increase temporal coherence is to provide the detector with additional image inputs from past frames. In CenterTrack, we provide the detection network with two frames as input: the current frame $I^{(t)}$ and the prior frame $I^{(t-1)}$. This allows the network to estimate the change in the scene and potentially recover occluded objects at time t from visual evidence at time $t-1$.

CenterTrack also takes prior detections $\{\mathbf{p}_0^{(t-1)}, \mathbf{p}_1^{(t-1)}, \ldots\}$ as additional input. How should these detections be represented in a form that is easily provided to a network? The point-based nature of our tracklets is helpful here. Since each detected object is represented by a single point, we can conveniently render all detections in a class-agnostic single-channel heatmap $H^{(t-1)} = \mathcal{R}(\{\mathbf{p}_0^{(t-1)}, \mathbf{p}_1^{(t-1)}, \ldots\})$, using the same Gaussian render function as in the training of point-based detectors. To reduce the propagation of false positive detections, we only render objects with a confidence score greater than a threshold τ. The architecture of CenterTrack is essentially identical to CenterNet, with four additional input channels. (See Figure 2.)

Tracking-conditioned detection provides a temporally coherent set of detected objects. However, it does not link these detections across time. In the next section, we show how to add one additional output to point-based detection to track objects through space and time.

4.2 Association through offsets

To associate detections through time, CenterTrack predicts a 2D displacement as two additional output channels $\hat{D}^{(t)} \in \mathbb{R}^{\frac{W}{R} \times \frac{H}{R} \times 2}$. For each detected object at location $\hat{\mathbf{p}}^{(t)}$, the displacement $\hat{\mathbf{d}}^{(t)} = \hat{D}^{(t)}_{\hat{\mathbf{p}}^{(t)}}$ captures the difference in location of the object in the current frame $\hat{\mathbf{p}}^{(t)}$ and the previous frame $\hat{\mathbf{p}}^{(t-1)}$: $\hat{\mathbf{d}}^{(t)} = \hat{\mathbf{p}}^{(t)} - \hat{\mathbf{p}}^{(t-1)}$. We learn this displacement using the same regression objective as size or location refinement:

$$L_{off} = \frac{1}{N} \sum_{i=1}^{N} \left| \hat{D}_{\mathbf{p}_i^{(t)}} - (\mathbf{p}_i^{(t-1)} - \mathbf{p}_i^{(t)}) \right|, \tag{3}$$

where $\mathbf{p}_i^{(t-1)}$ and $\mathbf{p}_i^{(t)}$ are tracked ground-truth objects. Figure 2 shows an example of this offset prediction.

With a sufficiently good offset prediction, a simple greedy matching algorithm can associate objects across time. For each detection at position $\hat{p}$, we greedily associate it with the closest unmatched prior detection at position $\hat{p} - \hat{D}_{\hat{p}}$, in descending order of confidence $\hat{w}$. If there is no unmatched prior detection within a radius κ, we spawn a new tracklet. We define κ as the geometric mean of the width and height of the predicted bounding box for each tracklet. A precise description of this greedy matching algorithm is provided in supplementary material. The simplicity of this greedy matching algorithm again highlights the advantages of tracking objects as points. A simple displacement prediction is sufficient to link objects across time. There is no need for a complicated distance metric or graph matching.

4.3 Training on video data

CenterTrack is first and foremost an object detector, and trained as such. The architectural changed from CenterNet to CenterTrack are minor: four additional input channels and two output channels. This allows us to fine-tune CenterTrack directly from a pre-trained CenterNet detector [56]. We copy all weights related to the current detection pipeline. All weights corresponding to additional inputs or outputs are initialized randomly. We follow the CenterNet training protocol and train all predictions as multi-task learning. We use the same training objective with the addition of offset regression L_{off}.

The main challenge in training CenterTrack comes in producing a realistic tracklet heatmap $H^{(t-1)}$. At inference time, this tracklet heatmap can contain an arbitrary number of missing tracklets, wrongly localized objects, or even false positives. These errors are not present in ground-truth tracklets $\{\mathbf{p}_0^{(t-1)}, \mathbf{p}_1^{(t-1)}, \ldots\}$ provided during training. We instead simulate this test-time error during training. Specifically, we simulate three types of error. First, we locally jitter each tracklet $\mathbf{p}^{(t-1)}$ from the prior frame by adding Gaussian noise to each center. That is, we render $p_i' = (x_i + r \times \lambda_{jt} \times w_i, y_i + r \times \lambda_{jt} \times h_i)$, where r is sampled from a Gaussian distribution. We use $\lambda_{jt} = 0.05$ in all experiments. Second, we randomly add false positives near ground-truth object locations by rendering a spurious noisy peak p_i' with probability λ_{fp}. Third, we simulate false negatives by randomly removing detections with probability λ_{fn}. λ_{fp} and λ_{fn} are set according to the statistics of our baseline model. These three augmentations are sufficient to train a robust tracking-conditioned object detector.

In practice, $I^{(t-1)}$ does not need to be the immediately preceding frame from time $t-1$. It can be a different frame from the same video sequence. In our experiments, we randomly sample frames near t to avoid overfitting to the framerate. Specifically, we sample from all frames k where $|k-t| < M_f$, where $M_f = 3$ is a hyperparameter.

4.4 Training on static image data

Without labeled video data, CenterTrack does not have access to a prior frame $I^{(t-1)}$ or tracked detections $\{\mathbf{p}_0^{(t-1)}, \mathbf{p}_1^{(t-1)}, \ldots\}$. However, we can simulate tracking on standard detection benchmarks, given only single images $I^{(t)}$ and detections $\{\mathbf{p}_0^{(t)}, \mathbf{p}_1^{(t)}, \ldots\}$. The idea is simple: we simulate the previous frame by randomly scaling and translating the current frame. As our experiments will demonstrate, this is surprisingly effective.

4.5 End-to-end 3D object tracking

To perform monocular 3D tracking, we adopt the monocular 3D detection form of CenterNet [56]. Specifically, we train output heads to predict object depth, rotation (encoded as an 8-dimensional vector [14]), and 3D extent. Since the projection of the center of the 3D bounding box may not align with the center of the object's 2D bounding box (due to perspective projection), we also predict a 2D-to-3D center offset. Further details are provided in the supplement.

5 Experiments

We evaluate 2D multi-object tracking on the MOT17 [28] and KITTI [12] tracking benchmarks. We also evaluate monocular 3D tracking on the nuScenes dataset [3]. Experiments on MOT16 can be found in the supplement.

5.1 Datasets and evaluation metrics

MOT. MOT17 contains 7 training sequences and 7 test sequences [28], The videos were captured by stationary cameras mounted in high-density scenes with heavy occlusion. Only pedestrians are annotated and evaluated. The video framerate is 25-30 FPS. The MOT dataset does not provide an official validation split. For ablation experiments, we split each training sequence into two halves, and use the first half frames for training and the second for validation. Our main results are reported on the test set.

KITTI. The KITTI tracking benchmark consists of 21 training sequences and 29 test sequences [12]. They are collected by a camera mounted on a car moving through traffic. The dataset provides 2D bounding box annotations for cars, pedestrians, and cyclists, but only cars are evaluated. Videos are captured at 10 FPS and contain large inter-frame motions. KITTI does not provide detections, and all entries use private detection. We again split all training sequences into halves for training and validation.

nuScenes. nuScenes is a newly released large-scale driving dataset with 7 object classes annotated for tracking [3]. It contains 700 training sequences, 150 validation sequences, and 150 test sequences. Each sequence contains roughly 40 frames at 2 FPS with 6 slightly overlapping images in a panoramic 360° view, resulting in 168k training, 36k validation, and 36k test images. The videos are sampled at 12 FPS, but frames are only annotated and evaluated at 2 FPS. All baselines and CenterTrack only use keyframes for training and evaluation. Due to the low framerate, the inter-frame motion is significant.

Evaluation metrics. We use the official evaluation metrics in each dataset. The common metric is multi-object tracking accuracy [24,40]: $MOTA = 1 - \frac{\sum_t (FP_t + FN_t + IDSW_t)}{\sum_t GT_t}$, where GT_t, FP_t, FN_t, and $IDSW_t$ are the number of ground-truth bounding boxes, false positives, false negatives, and identity switches in frame t, respectively. MOTA does not rank tracklets according to confidence and is sensitive to the task-dependent output threshold θ [46]. The thresholds we use are listed in Section 5.2. The interplay between output threshold and true positive criteria matters. For 2D tracking [12, 28], > 0.5 bounding box IoU is a the true positive. For 3D tracking [3], bounding box center distance $< 2m$ on the ground plane is the criterion for a true positive. When objects are successfully detected, but not tracked, they are identified as an identity switch (IDSW). The IDF1 metric measures the minimal cost change from predicted ids to the correct ids. In our ablation studies, we report false positve rate (FP) $\frac{\sum_t FP_t}{\sum_t GT_t}$, false negative rate (FN) $\frac{\sum_t FN_t}{\sum_t GT_t}$, and identity switches (IDSW) $\frac{\sum_t IDSW_t}{\sum_t GT_t}$ separately. In comparisons with other methods, we report the absolute numbers following the dataset convention [12, 28]. We also report the Most Tracked ratio (MT) for the ratio of most tracked ($> 80\%$ time) objects and Most Lost ratio (ML) for most lost ($< 20\%$ time) objects [40].

nuScenes adopts a more robust metric, AMOTA, which is a weighted average of MOTA across different output thresholds. Specifically,

$$AMOTA = \frac{1}{n-1} \sum_{r \in \{\frac{1}{n-1}, \frac{2}{n-1}, \cdots, 1\}} MOTA_r$$

$$MOTA_r = max(0, 1 - \alpha \frac{IDSW_r + FP_r + FN_r - (1-r) \times P}{r \times P})$$

where r is a fixed recall threshold, $P = \sum_t GT_t$ is the total number of annotated objects among all frames, and $FP_r = \sum_t FP_{r,t}$ is the total number of false positive samples only considering the top confident samples that achieve the recall threshold r. The hyperparameters $n = 40$ and $\alpha = 0.2$ (AMOTA@0.2), or $\alpha = 1$ (AMOTA@1) are set by the benchmark organizers. The overall AMOTA is the average AMOTA among all 7 categories.

5.2 Implementation details

Our implementation is based on CenterNet [56]. We use DLA [53] as the network backbone, optimized with Adam [20] with learning rate $1.25e-4$ and batchsize 32. Data augmentations include random horizontal flipping, random resized cropping, and color jittering. For all experiments, we train the networks for 70 epochs. The learning rate is dropped by a factor of 10 at the 60th epoch. We test the runtime on a machine with an Intel Core i7-8086K CPU and a Titan Xp GPU. The runtimes depend on the number of objects for rendering and the input resolution in each dataset.

The MOT dataset [28] annotates each pedestrian as an amodal bounding box. That is, the bounding box always covers the whole body even when part of the object is out of the frame. In contrast, CenterNet [56] requires the center of each inferred bounding box to be within the frame. To handle this, we separately predict the visible and amodal bounding boxes [42]. Further details on this can be found in the supplement. We follow prior works [32, 39, 41, 52, 55] to pretrain on external data. We train our network on the CrowdHuman [34] dataset, using the static image training described in Section 4.4. Details on the CrowdHuman dataset and ablations of pretraining are in the supplement.

The default input resolution for MOT images is 1920×1080. We resize and pad the images to 960×544. We use random false positive ratio $\lambda_{fp} = 0.1$ and random false negative ratio $\lambda_{fn} = 0.4$. We only output tracklets that have a confidence of $\theta = 0.4$ or higher, and set the heatmap rendering threshold to $\tau = 0.5$. A controlled study of these hyperparameters is in the supplement.

For KITTI [12], we keep the original input resolution 1280×384 in training and testing. The hyperparameters are set at $\lambda_{fp} = 0.1$ and $\lambda_{fn} = 0.2$, with output threshold $\theta = 0.4$ and rendering threshold $\tau = 0.4$. We fine-tune our KITTI model from a nuScenes tracking model.

For nuScenes [3], we use input resolution 800×448. We set $\lambda_{fp} = 0.1$ and $\lambda_{fn} = 0.4$, and use output threshold $\theta = 0.1$ and rendering threshold $\tau = 0.1$. We first train our nuScenes model for 140 epochs for just 3D detection [56] and then fine-tune for 70 epochs for 3D tracking. Note that nuScenes evaluation is done per 360 panorama, not per image. We naively fuse all outputs from the 6 cameras together, without handling duplicate detections at the intersection of views [38].

Track rebirth. Following common practice [1, 55], we keep unmatched tracks "inactive" until they remain undetected for K consecutive frames. Inactive tracks can be matched to detections and regain their ID, but not appear in the prior heatmap or output. The tracker stays online. Rebirth only matters for the MOT test set, where we use $K = 32$. For all other experiments, we found rebirth not to be required ($K = 0$).

	Time(ms)	MOTA ↑	IDF1 ↑	MT ↑	ML ↓	FP ↓	FN ↓	IDSW ↓
Tracktor17 [1]	666+D	53.5	52.3	19.5	36.6	12201	248047	2072
LSST17 [10]	666+D	54.7	**62.3**	20.4	40.1	26091	228434	**1243**
Tracktor v2 [1]	666+D	56.5	55.1	21.1	35.3	**8866**	235449	3763
GMOT	167+D	55.4	57.9	22.7	34.7	20608	229511	1403
Ours (Public)	**57+D**	**61.5**	59.6	**26.4**	**31.9**	14076	**200672**	2583
Ours (Private)	57	67.8	64.7	34.6	24.6	18498	160332	3039

Table 1: Evaluation on the MOT17 test sets (top: public detection; bottom: private detection). We compare to published entries on the leaderboard. The runtime is calculated from the HZ column on the leaderboard. +D means detection time, which is usually $>$ 100ms [31].

5.3 Public detection

The MOT17 challenge only supports public detection. That is, participants are asked to use the provided detections. Public detection is meant to test a tracker's ability to associate objects, irrespective of its ability to detect objects. Our method operates in the private detection mode by default. For the MOT challenge we created a public-detection version of CenterTrack that uses the externally provided (public) detections and is thus fairly compared to other participants in the challenge. This shows that the advantages of CenterTrack are not due to the accuracy of the detections but are due to the tracking framework itself.

Note that refining and rescoring the given bounding boxes is allowed and is commonly used by participants in the challenge [1, 19, 26]. Following Tracktor [1], we keep the bounding boxes that are close to an existing bounding box in the previous frame. We only initialize a new trajectory if it is near a public detection. All bounding boxes in our results are either near a public detection in the current frame or near a tracked box in the previous frame. The algorithm's diagram of this public-detection configuration can be found in the supplement. We use this public-detection configuration of CenterTrack for MOT17 test set evaluation and use the private-detection setting in our ablation studies.

5.4 Main results

All three datasets – MOT17 [28], KITTI [12], and nuScenes [3] – host test servers with hidden annotations and leaderboards. We compare to all published results on these leaderboards. The numbers were accessed on Mar. 5th, 2020. We retrain CenterTrack on the full training set with the same hyperparameters in the ablation experiments.

Table 1 lists the results on the MOT17 challenge. We use our public configuration in Section 5.3 and do not pretrain on CrowdHuman [34]. CenterTrack significantly outperforms the prior state of the art even when restricted to the public-detection configuration. For example CenterTrack improves MOTA by 5 points (an 8.6% relative improvement) over Tracktor v2 [1].

The public detection setting ensures that all methods build on the same underlying detector. Our gains come from two sources. Firstly, the heatmap input makes our tracker better preserve tracklets from the previous frame, which results in a much lower rate of

	Time(ms)	MOTA ↑	MOTP ↑	MT ↑	ML ↓	IDSW ↓	FRAG ↓
AB3D [46]	4+D	83.84	85.24	66.92	11.38	**9**	**224**
BeyondPixel [35]	300+D	84.24	85.73	73.23	2.77	468	944
3DT [14]	30+D	84.52	85.64	73.38	2.77	377	847
mmMOT [54]	10+D	84.77	85.21	73.23	2.77	284	753
MOTSFusion [27]	440+D	84.83	85.21	3.08	2.77	275	759
MASS [18]	10+D	85.04	**85.53**	74.31	2.77	301	744
Ours	82	**89.44**	85.05	**82.31**	**2.31**	116	334

Table 2: Evaluation on the KITTI test set. We compare to all published entries on the leaderboard. Runtimes are from the leaderboard. +D means detection time.

	Time(ms)	AMOTA@0.2 ↑	AMOTA@1 ↑	AMOTP ↓
Mapillary [38]+AB3D [46]	-	6.9	1.8	1.8
Ours	45	**27.8**	**4.6**	**1.5**

Table 3: Evaluation on the nuScenes test set. We compare to the official monocular 3D tracking baseline, which applies a state-of-the-art 3D tracker [46]. We list the average AMOTA@0.2, AMOTA@1, and AMOTP over all 7 categories.

false negatives. And second, our simple learned offset is effective. (See Section 5.6 for more analysis.) For reference, we also included a private detection version, where CenterTrack simultaneously detects and tracks objects (Table 1, bottom). It further improves the MOTA to 67.3%, and runs at 17 FPS end-to-end (including detection).

For IDF1 and id-switch, our local model is not as strong as offline methods such as LSST17 [10], but is better than other online methods [1]. We believe that there is an exciting avenue for future work in combining local trackers (such as our work) with stronger offline long-range models (such as SORT [2], LMP [41], and other ReID-based trackers [50, 52]).

On KITTI [12], we submitted our best-performing model with flip testing [56]. The model runs at 82ms and yields 89.44% MOTA, outperforming all published work (Table 2). Note that our model without flip testing runs at 45ms with 88.7% MOTA on the validation set (vs. 89.63% with flip testing on the validation set). We avoid submitting to the test server multiple times following their test policy. The results again indicate that CenterTrack performs competitively with more complex methods.

On nuScenes [3], our monocular tracking method achieves an AMOTA@0.2 of 28.3% and an AMOTA@1 of 4.6%, outperforming the monocular baseline [38, 46] by a large margin. There are two main reasons. Firstly, we use a stronger and faster 3D detector [56] (see the 3D detector comparison in the supplementary). More importantly, as shown in Table 6, the Kalman-filter-based 3D tracking baseline relies on hand-crafted motion rules [46], which are less effective in low-framerate regimes. Our method learns object motion from data and is much more stable at low framerates.

	MOT17				KITTI				nuScenes	
	MOTA↑	FP↓	FN↓	IDSW↓	MOTA↑	FP↓	FN↓	IDSW↓	AMOTA@0.2↑	AMOTA@1↑
detection only	63.6	3.5%	30.3%	2.5 %	84.3	4.3%	9.8%	1.5%	18.1	3.4
w/o offset	65.8	**4.5%**	**28.4%**	1.3%	87.1	**5.4%**	**5.8%**	1.6%	17.8	3.6
w/o heatmap	63.9	3.5%	30.3%	2.3%	85.4	4.3%	9.8%	0.4%	26.5	5.9
Ours	**66.1**	**4.5%**	**28.4%**	**1.0%**	**88.7**	**5.4%**	**5.8%**	**0.1%**	**28.3**	**6.8**

Table 4: Ablation study on MOT17, KITTI, and nuScenes. All results are on validation sets (Section 5.1). For each dataset, we report the corresponding official metrics. ↑ indicates that higher is better, ↓ indicates that lower is better.

5.5 Ablation studies

We first ablate our two main technical contributions: tracking-conditioned detection (Section 4.1) and offset prediction (Section 4.2) on all three datasets. Specifically, we compare our full framework with three baselines.

Detection only runs a CenterNet detector at each individual frame and associates their identity only based on 2D center distance. This model does not use video data, but still uses two input images.

Without offset uses just tracking-conditioned prediction with a predicted offset of zero. Every object is again associated to its closest object in the previous frame.

Without heatmap predicts the center offset between frames and uses the updated center distance as the association metric, but the prior heatmap is not provided. The offset-based greedy association is used.

Table 4 shows the results. On all datasets, our full CenterTrack model performs significantly better than the baselines. Tracking-conditioned detection yields $\sim 2\%$ MOTA improvement on MOT and $\sim 3\%$ MOTA improvement on KITTI, with or without offset prediction. It produces more false positives but fewer false negatives. This is because with the heatmap prior, the network tends to predict more objects around the previous peaks, which are sometimes misleading. The merits of the heatmap outweigh the limitations and improve MOTA overall. Using the prior heatmap also significantly reduces IDSW on both datasets, indicating that the heatmap stabilizes detection.

Tracking offset prediction gives a huge boost on nuScenes and reduces IDSW consistently in MOT and KITTI. The effectiveness of the tracking offset appears to be related to the video framerate. When the framerate is high, motion between frames is small, and a zero offset is often a reasonable starting point for association. When framerate is low, as in the nuScenes dataset, motion between frames is large and static object association is considerably less effective. Our offset prediction scheme helps deal with such large inter-frame motion. Next, we ablate other components on MOT17.

Training with noisy heatmap. The 2nd row in Table 5 shows the importance of injecting noise into heatmaps during training (Section 4.3). Without noise injection, the model fails to generalize and yields dramatically lower accuracy. In particular, this model has a large false negative rate. One reason is that in the first frame, the input heatmap is empty. This model had a hard time discovering new objects that were not indicated in the prior heatmap.

Training on static images. We train a version of our model on static images only, as described in Section 4.4. The results are shown in Table 5 (3rd row, 'Static image'). As

	MOTA ↑	IDF1 ↑	MT ↑	ML ↓	FP ↓	FN ↓	IDSW ↓
Ours	66.1	64.2	41.3	21.2	4.5%	28.4%	1.0%
w.o. noisy hm	34.4	46.2	26.3	42.2	7.3%	57.4%	0.9%
Static image	66.1	65.4	41.6	19.2	5.4%	27.5%	1.0%
w. Hungarian	66.1	61.0	40.7	20.9	4.5%	28.3%	1.0%
w. rebirth	66.2	69.4	39.5	22.1	3.9%	29.5%	0.4%

Table 5: Additional experiments on the MOT17 validation set. From top to bottom: our model, our model trained without simulating heatmap noise, our model trained on static images only, our model with Hungarian matching, and our model with track rebirth.

	MOT17				KITTI				nuScenes	
	MOTA↑	FP↓	FN↓	IDSW↓	MOTA↑	FP↓	FN↓	IDSW↓	AMOTA@0.2↑	AMOTA@1↑
no motion	65.8	**4.5%**	**28.4%**	1.3%	87.1	**5.4%**	**5.8%**	1.6%	17.8	3.6
Kalman filter	**66.1**	**4.5%**	**28.4%**	**1.0%**	87.9	**5.4%**	**5.8%**	0.9%	18.3	3.8
optical flow	**66.1**	**4.5%**	**28.4%**	**1.0%**	88.4	**5.4%**	**5.8%**	0.4%	26.6	6.2
ours	**66.1**	**4.5%**	**28.4%**	**1.0%**	**88.7**	**5.4%**	**5.8%**	**0.1%**	**28.3**	**6.8**

Table 6: Comparing different motion models on MOT17, KITTI, and nuScenes. All results are on validation sets (Section 5.1). All experiments on the same dataset are from the same model.

reported in this table, training on static images gives the same performance as training on videos on the MOT dataset. Separately, we observed that training on static images is less effective on nuScenes, where framerate is low.

Matching algorithm. We use a simple greedy matching algorithm based on the detection score, while most other trackers use the Hungarian algorithm. We show the performance of CenterTrack with Hungarian matching in the 4th row of Table 5. It does not improve performance. We choose greedy matching for simplicity.

Track rebirth. We show CenterTrack with track rebirth (K=32) in the last row of Table 5. While the MOTA performance keeps similar, it significantly increases IDF1 and reduces ID switch. We use this setting for our MOT test set submission. For other datasets and evaluation metrics no rebirth was required ($K = 0$).

5.6 Comparison to alternative motion models

Our offset prediction is able to estimate object motion, but also performs a simple association, as current objects are linked to prior detections, which CenterTrack receives as one of its inputs. To verify the effectiveness of our learned association, we replace our offset prediction with three alternative motion models:

No motion. We set the offset to zeros. It is copied from Table 4 for reference only.

Kalman filter. The Kalman filter predicts each object's future state through an explicit motion model estimated from its history. It is the most widely used motion model in traditional real-time trackers [2,46,47]. We use the popular public implementation from SORT [2].

Optical flow. As an alternative motion model, we use FlowNet2 [15]. The model was trained to estimate dense pixel motion for all objects in a scene. We run the strongest officially released FlowNet2 model ($\sim$ 150ms / image pair), and replace our learned offset with the predicted optical flow at each predicted object center.

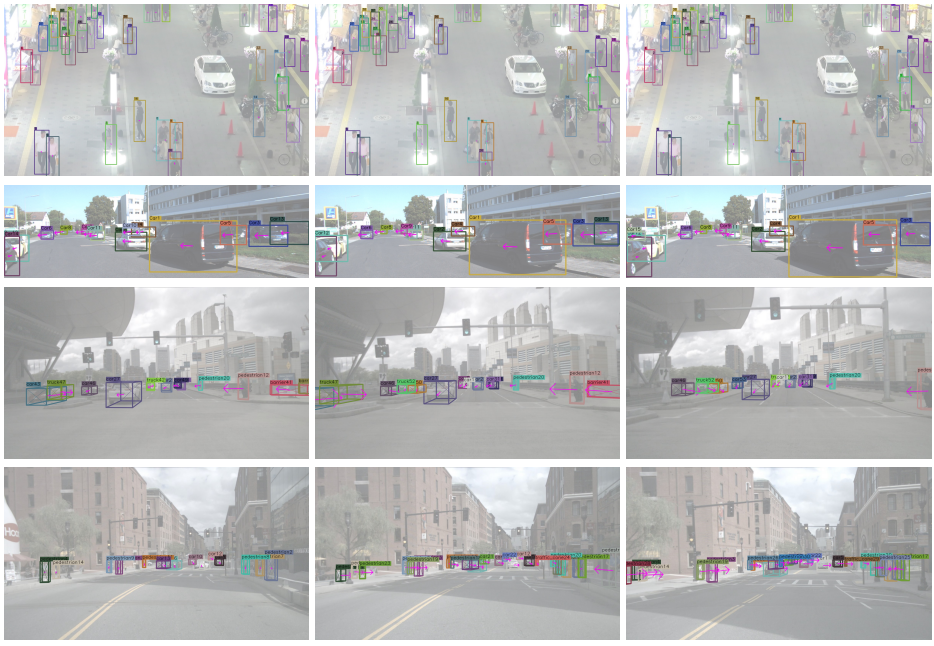

Fig. 3: Qualitative results on MOT (1st row), KITTI (2nd row), and nuScenes (3rd and 4th rows). Each row shows three consecutive frames. We show the predicted tracking offset in arrow. Tracks are coded by color. Best viewed on the screen.

The results are shown in Table 6. All models use the exact same detector. On the high-framerate MOT17 dataset, any motion model suffices, and even no motion model at all performs competitively. On KITTI and nuScenes, where the intra-frame motions are non-trivial, the hand-crafted motion rule of the Kalman filter performs significantly worse, and even the performance of optical flow degrades. This emphasizes that our offset model does more than just motion estimation. CenterTrack is conditioned on prior detections and can learn to snap offset predictions to exactly those prior detections. Our training procedure strongly encourages this through heavy data augmentation.

6 Conclusion

We presented an end-to-end simultaneous object detection and tracking framework. Our method takes two frames and a prior heatmap as input, and produces detection and tracking offsets for the current frame. Our tracker is purely local and associates objects greedily through time. It runs online (no knowledge of future frames) and in real time, and sets a new state of the art on the challenging MOT17, KITTI, and nuScenes 3D tracking benchmarks.

Acknowledgements. This work has been supported in part by the National Science Foundation under grant IIS-1845485.

References

1. Bergmann, P., Meinhardt, T., Leal-Taixe, L.: Tracking without bells and whistles. In: ICCV (2019)
2. Bewley, A., Ge, Z., Ott, L., Ramos, F., Upcroft, B.: Simple online and realtime tracking. In: ICIP (2016)
3. Caesar, H., Bankiti, V., Lang, A.H., Vora, S., Liong, V.E., Xu, Q., Krishnan, A., Pan, Y., Baldan, G., Beijbom, O.: nuScenes: A multimodal dataset for autonomous driving. In: CVPR (2020)
4. Carreira, J., Agrawal, P., Fragkiadaki, K., Malik, J.: Human pose estimation with iterative error feedback. In: CVPR (2016)
5. Choi, W., Savarese, S.: Multiple target tracking in world coordinate with single, minimally calibrated camera. In: ECCV (2010)
6. Evangelidis, G.D., Psarakis, E.Z.: Parametric image alignment using enhanced correlation coefficient maximization. TPAMI (2008)
7. Fang, K., Xiang, Y., Li, X., Savarese, S.: Recurrent autoregressive networks for online multi-object tracking. In: WACV (2018)
8. Feichtenhofer, C., Pinz, A., Zisserman, A.: Detect to track and track to detect. In: ICCV (2017)
9. Felzenszwalb, P.F., Girshick, R.B., McAllester, D., Ramanan, D.: Object detection with discriminatively trained part-based models. In: TPAMI (2009)
10. Feng, W., Hu, Z., Wu, W., Yan, J., Ouyang, W.: Multi-object tracking with multiple cues and switcher-aware classification. arXiv:1901.06129 (2019)
11. Fieraru, M., Khoreva, A., Pishchulin, L., Schiele, B.: Learning to refine human pose estimation. In: CVPR Workshops (2018)
12. Geiger, A., Lenz, P., Urtasun, R.: Are we ready for autonomous driving? The KITTI vision benchmark suite. In: CVPR (2012)
13. Held, D., Thrun, S., Savarese, S.: Learning to track at 100 fps with deep regression networks. In: ECCV (2016)
14. Hu, H.N., Cai, Q.Z., Wang, D., Lin, J., Sun, M., Krhenbhl, P., Darrell, T., Yu, F.: Joint monocular 3D detection and tracking. In: ICCV (2019)
15. Ilg, E., Mayer, N., Saikia, T., Keuper, M., Dosovitskiy, A., Brox, T.: Flownet 2.0: Evolution of optical flow estimation with deep networks. In: CVPR (2017)
16. Kang, K., Li, H., Xiao, T., Ouyang, W., Yan, J., Liu, X., Wang, X.: Object detection in videos with tubelet proposal networks. In: CVPR (2017)
17. Kang, K., Li, H., Yan, J., Zeng, X., Yang, B., Xiao, T., Zhang, C., Wang, Z., Wang, R., Wang, X., et al.: T-cnn: Tubelets with convolutional neural networks for object detection from videos. Circuits and Systems for Video Technology (2017)
18. Karunasekera, H., Wang, H., Zhang, H.: Multiple object tracking with attention to appearance, structure, motion and size. IEEE Access (2019)
19. Keuper, M., Tang, S., Andres, B., Brox, T., Schiele, B.: Motion segmentation & multiple object tracking by correlation co-clustering. TPAMI (2018)
20. Kingma, D.P., Ba, J.: Adam: A method for stochastic optimization. ICLR (2015)
21. Lang, A.H., Vora, S., Caesar, H., Zhou, L., Yang, J., Beijbom, O.: Pointpillars: Fast encoders for object detection from point clouds. In: CVPR (2019)
22. Law, H., Deng, J.: Cornernet: Detecting objects as paired keypoints. In: ECCV (2018)
23. Leal-Taixé, L., Canton-Ferrer, C., Schindler, K.: Learning by tracking: Siamese cnn for robust target association. In: CVPR Workshops (2016)
24. Leal-Taixé, L., Milan, A., Schindler, K., Cremers, D., Reid, I., Roth, S.: Tracking the trackers: an analysis of the state of the art in multiple object tracking. arXiv:1704.02781 (2017)

25. Lin, T.Y., Goyal, P., Girshick, R., He, K., Dollár, P.: Focal loss for dense object detection. In: ICCV (2017)
26. Long, C., Haizhou, A., Zijie, Z., Chong, S.: Real-time multiple people tracking with deeply learned candidate selection and person re-identification. In: ICME (2018)
27. Luiten, J., Fischer, T., Leibe, B.: Track to reconstruct and reconstruct to track. arXiv:1910.00130 (2019)
28. Milan, A., Leal-Taixé, L., Reid, I., Roth, S., Schindler, K.: MOT16: A benchmark for multi-object tracking. arXiv:1603.00831 (2016)
29. Moon, G., Chang, J., Lee, K.M.: Posefix: Model-agnostic general human pose refinement network. In: CVPR (2019)
30. Ren, J., Chen, X., Liu, J., Sun, W., Pang, J., Yan, Q., Tai, Y.W., Xu, L.: Accurate single stage detector using recurrent rolling convolution. In: CVPR (2017)
31. Ren, S., He, K., Girshick, R., Sun, J.: Faster R-CNN: Towards real-time object detection with region proposal networks. In: NIPS (2015)
32. Sadeghian, A., Alahi, A., Savarese, S.: Tracking the untrackable: Learning to track multiple cues with long-term dependencies. In: ICCV (2017)
33. Schulter, S., Vernaza, P., Choi, W., Chandraker, M.: Deep network flow for multi-object tracking. In: CVPR (2017)
34. Shao, S., Zhao, Z., Li, B., Xiao, T., Yu, G., Zhang, X., Sun, J.: Crowdhuman: A benchmark for detecting human in a crowd. arXiv:1805.00123 (2018)
35. Sharma, S., Ansari, J.A., Murthy, J.K., Krishna, K.M.: Beyond pixels: Leveraging geometry and shape cues for online multi-object tracking. In: ICRA (2018)
36. Shi, J., Tomasi, C.: Good features to track. In: CVPR (1994)
37. Shi, S., Wang, X., Li, H.: Pointrcnn: 3D object proposal generation and detection from point cloud. In: CVPR (2019)
38. Simonelli, A., Bulò, S.R.R., Porzi, L., López-Antequera, M., Kontschieder, P.: Disentangling monocular 3d object detection. In: ICCV (2019)
39. Son, J., Baek, M., Cho, M., Han, B.: Multi-object tracking with quadruplet convolutional neural networks. In: CVPR (2017)
40. Stiefelhagen, R., Bernardin, K., Bowers, R., Garofolo, J., Mostefa, D., Soundararajan, P.: The CLEAR 2006 evaluation. In: CLEAR (2006)
41. Tang, S., Andriluka, M., Andres, B., Schiele, B.: Multiple people tracking by lifted multicut and person re-identification. In: CVPR (2017)
42. Tian, Z., Shen, C., Chen, H., He, T.: FCOS: Fully convolutional one-stage object detection. In: ICCV (2019)
43. Tomasi, C., Kanade, T.: Detection and tracking of point features. Technical Report CMU-CS-91-132, Carnegie Mellon University (1991)
44. Tu, Z.: Auto-context and its application to high-level vision tasks. In: CVPR (2008)
45. Voigtlaender, P., Krause, M., Osep, A., Luiten, J., Sekar, B.B.G., Geiger, A., Leibe, B.: Mots: Multi-object tracking and segmentation. In: CVPR (2019)
46. Weng, X., Kitani, K.: A baseline for 3d multi-object tracking. arXiv:1907.03961 (2019)
47. Wojke, N., Bewley, A., Paulus, D.: Simple online and realtime tracking with a deep association metric. In: ICIP (2017)
48. Xiang, Y., Alahi, A., Savarese, S.: Learning to track: Online multi-object tracking by decision making. In: ICCV (2015)
49. Xiao, B., Wu, H., Wei, Y.: Simple baselines for human pose estimation and tracking. In: ECCV (2018)
50. Xu, J., Cao, Y., Zhang, Z., Hu, H.: Spatial-temporal relation networks for multi-object tracking. In: ICCV (2019)
51. Yang, F., Choi, W., Lin, Y.: Exploit all the layers: Fast and accurate cnn object detector with scale dependent pooling and cascaded rejection classifiers. In: CVPR (2016)

52. Yu, F., Li, W., Li, Q., Liu, Y., Shi, X., Yan, J.: Poi: Multiple object tracking with high performance detection and appearance feature. In: ECCV Workshops (2016)
53. Yu, F., Wang, D., Shelhamer, E., Darrell, T.: Deep layer aggregation. In: CVPR (2018)
54. Zhang, W., Zhou, H., Sun, S., Wang, Z., Shi, J., Loy, C.C.: Robust multi-modality multi-object tracking. In: ICCV (2019)
55. Zhang, Z., Cheng, D., Zhu, X., Lin, S., Dai, J.: Integrated object detection and tracking with tracklet-conditioned detection. arXiv:1811.11167 (2018)
56. Zhou, X., Wang, D., Krähenbühl, P.: Objects as points. arXiv:1904.07850 (2019)
57. Zhu, B., Jiang, Z., Zhou, X., Li, Z., Yu, G.: Class-balanced grouping and sampling for point cloud 3D object detection. arXiv:1908.09492 (2019)
58. Zhu, J., Yang, H., Liu, N., Kim, M., Zhang, W., Yang, M.H.: Online multi-object tracking with dual matching attention networks. In: ECCV (2018)
59. Zhu, X., Wang, Y., Dai, J., Yuan, L., Wei, Y.: Flow-guided feature aggregation for video object detection. In: ICCV (2017)

A Tracking algorithms

A.1 Private tracking

We adopt a simple greedy id association algorithm based on the center distance, shown in Algorithm 1. We use the same algorithm for both 2D tracking and 3D tracking.

A.2 Public tracking

For public tracking, we follow Tractor [1] to extend a private tracking algorithm to public detection. The id association is exactly the same as private detection (Line 1 to Line 14). The difference lies in how a track can be created. In public detection, we only initialize a track if it is near a provided bounding box (Line 17 to Line 21).

B Results on MOT16

MOT16 shares the same training and testing sequences with MOT17, but officially supports private detection. As is shown in Table 7, we rank 2nd among all published entries. We remark that all other entries use a heavy detector trained on private data [52] and many rely on slow matching schemes [41, 52]. For example, LMP_p [41] computes person-reidentification features for all pairs of bounding boxes using a Siamese network, requiring $O(n^2)$ forward passes through a deep network. In contrast, CenterTrack involves a single pass through a network and operates online at 17 FPS.

	Time(ms)	MOTA ↑	IDF1 ↑	FP ↓	FN ↓	IDSW ↓
SORT [2]	36+D	60.4	56.1	11183	59867	1135
DeepSORT [47]	59+D	61.4	62.2	12852	56668	781
POI [52]	100+D	66.1	65.1	5061	55914	805
KNDT [52]	1428+D	68.2	60.0	11479	45605	933
LMP_p [41]	2000+D	**71.0**	**70.1**	**7880**	44564	**434**
Ours (Private)	**57**	69.6	60.7	10458	**42805**	2124

Table 7: Evaluation on the MOT16 test sets (private detection). We compare to all published on the leaderboard. The runtime is calculated from the HZ column on the leaderboard. +D means detection time, which is usually $>$ 100ms [31].

Algorithm 1: Private Detection

Input : $T^{(t-1)} = \{(\mathbf{p}, \mathbf{s}, id)_j^{(t-1)}\}_{j=1}^{M}$: Tracked objects in the previous frame, with center $\mathbf{p}$, size $\mathbf{s} = (w, h)$.
$\hat{B}^{(t)} = \{(\hat{\mathbf{p}}, \hat{\mathbf{d}})_i^{(t)}\}_{i=1}^{N}$: Heatmap peaks with offset $\hat{\mathbf{d}}$ in the current frame, sorted in desending confidence.

Output: $T^{(t)} = \{(\mathbf{p}, \mathbf{s}, id)_i^{(t)}\}_{i=1}^{N}$: Tracked objects in the current frame.

// **Initialization:** $T^{(t)}$ and $\mathcal{S}$ are initialized as empty lists.
$T^{(t)} \leftarrow \emptyset$
$S \leftarrow \emptyset$ // Set of matched tracks
$W \leftarrow Cost(B^{(t)}, T^{(t-1)})$// $W_{ij} = ||\hat{\mathbf{p}}_i^{(t)} - \hat{\mathbf{d}}_i^{(t)}, \mathbf{p}_j^{(t-1)}||_2$

for $i \leftarrow 1$ *to* N **do**
  $j \leftarrow \arg\min_{j \notin S} W_{ij}$
  // calculate the distance threshold κ
  $\kappa \leftarrow \min(\sqrt{\hat{w}_i \hat{h}_i}, \sqrt{w_j h_j})$
  // if the cost is smaller the threshold.
  if $w_{ij} < \kappa$ **then**
    // Propagate matched id
    $T^{(t)} \leftarrow T^{(t)} \cup (\hat{\mathbf{p}}_i^{(t)}, \hat{\mathbf{s}}_i^{(t)}, id_j^{(t-1)})$
    $S \leftarrow S \cup \{j\}$ // Mark track j as matched
  end
  else
    // Create a new track.
    $T^{(t)} \leftarrow T^{(t)} \cup (\hat{\mathbf{p}}_i^{(t)}, \hat{\mathbf{s}}_i^{(t)}, NewId)$
  end
end
Return: $T^{(t)}$

Algorithm 2: Public Detection

Input : $T^{(t-1)} = \{(\mathbf{p}, \mathbf{s}, id)_j^{(t-1)}\}_{j=1}^{M}$: Tracked objects in the previous frame, with center $\mathbf{p}$, size $\mathbf{s} = (w, h)$.
$\hat{B}^{(t)} = \{(\hat{\mathbf{p}}, \hat{\mathbf{d}})_i^{(t)}\}_{i=1}^{N}$: Heatmap peaks with offset $\hat{\mathbf{d}}$ in the current frame, sorted in desending confidence.
$\hat{D}^{(t)} = \{(\mathbf{p}, \mathbf{s})_k^{(t)}\}_{k=1}^{K}$: Public detections.

Output: $T^{(t)} = \{(\mathbf{p}, \mathbf{s}, id)_{i'}^{(t)}\}_{i'=1}^{N'}$: Tracked objects in the current frame.

// **Initialization:** $T^{(t)}$ and $\mathcal{S}$ are initialized as empty lists.
$T^{(t)} \leftarrow \emptyset$
$S \leftarrow \emptyset$ // Set of matched tracks
$W \leftarrow Cost(B^{(t)}, T^{(t-1)})$// $W_{ij} = ||\hat{\mathbf{p}}_i^{(t)} - \hat{\mathbf{d}}_i^{(t)}, \mathbf{p}_j^{(t-1)}||_2$
$W' \leftarrow Cost(B^{(t)}, D^{(t)})$// $W'_{ik} = ||\hat{\mathbf{p}}_i^{(t)}, \mathbf{p}_k^{(t)}||_2$

for $i \leftarrow 1$ *to* N **do**
  $j \leftarrow \arg\min_{j \notin S} W_{ij}$
  // calculate the distance threshold κ
  $\kappa \leftarrow \min(\sqrt{\hat{w}_i \hat{h}_i}, \sqrt{w_j h_j})$
  // if the cost is smaller the threshold.
  if $w_{ij} < \kappa$ **then**
    // Propagate matched id
    $T^{(t)} \leftarrow T^{(t)} \cup (\hat{\mathbf{p}}_i^{(t)}, \hat{\mathbf{s}}_i^{(t)}, id_j^{(t-1)})$
    $S \leftarrow S \cup \{j\}$ // Mark track j as matched
  end
  else
    $k \leftarrow \arg\min_{k=1}^{K} \leftarrow W'_{ik}$
    $\kappa' \leftarrow \min(\sqrt{\hat{w}_i \hat{h}_i}, \sqrt{w_k h_k})$
    if $W'_{ik} < \kappa'$ **then**
      // Create a new track.
      $T^{(t)} \leftarrow T^{(t)} \cup (\hat{\mathbf{p}}_i^{(t)}, \hat{\mathbf{s}}_i^{(t)}, NewId)$
    end
  end
end
Return: $T^{(t)}$

	Modality	mAP ↑	mATE ↓	mASE ↓	mAOE ↓	mAVE ↓	mAAE ↓	NDS ↑
Megvii [57]	LiDAR	52.8	0.300	0.247	0.379	0.245	0.140	63.3
PointPillars [21]	LiDAR	30.5	0.517	0.290	0.500	0.316	0.368	45.3
Mappilary [38]	Camera	30.4	0.738	0.263	0.546	1.	0.134	38.4
CenterNet [56]	Camera	33.8	0.658	0.255	0.629	1.	0.141	40.1

Table 8: 3D detection results on nuScenes test set. We show 3D bounding box mAP, mean translation error (mATE), mean size error (mASE), mean orientation error (mAOE), mean velocity error (mATE), mean attributes error (mAAE), and their weighted (with weight 5 on mAP and 1 on others) average NDS.

C 3D detection

We follow CenterNet [56] to regress to object depth $\hat{D} \in R^{\frac{W}{R} \times \frac{H}{R}}$, 3d extent $\hat{\Gamma} \in R^{\frac{W}{R} \times \frac{H}{R} \times 3}$, orientation (encoded as an 8-dimension vector) $\hat{A} \in R^{\frac{W}{R} \times \frac{H}{R} \times 8}$. The training loss for these are identical to CenterNet [56]. Since the 2D bounding box center does not align with the projected 3D bounding box center due to perspective projection, we in addition regress to an offset from the 2D center to the projected 3D bounding box center$\hat{F} \in R^{\frac{W}{R} \times \frac{H}{R} \times 2}$. We use L1Loss:

$$L_{off3d} = \frac{1}{N} \sum_{k=1}^{N} |\hat{f}_k - f_k|, \tag{4}$$

where $f_k \in \mathcal{R}^2$ is the ground truth offset of object k, and $\hat{f}_k = \hat{F}_{\mathbf{p}_k}$ is the value in $\hat{F}$ at location $\mathbf{p}_k$.

We show the 3D detection performance of CenterNet [56] with the offset prediction in Table 8 for reference. The 3D detection performance is on-par with Mappilary [38] and PointPillars [21], but far below the LiDAR based state-of-the-art Megvii [57].

D Amodal bounding box regression

CenterNet [56] requires the bounding box center to be within the image. While in MOT [28], the center of the annotated bounding box (Amodal bounding box) can be outside of the image. To accommodate this case, We extend the 2-channel bounding box size head in CenterNet to a 4-channel head $\hat{A} \in \mathbb{R}^{\frac{W}{R} \times \frac{H}{R} \times 4}$ for the distance to the top-, left-, bottom-, right-bounding box border. Note that we still detect the in-frame bounding box center and regress to the in-frame bounding box size. With this 4-dimensional bounding box formulation, the output bounding box is not necessarily centered on the detected center. The training loss for the 4-dimensional bounding box formulation is L1Loss:

$$L_{amodal\_size} = \frac{1}{N} \sum_{i=1}^{N} |\hat{A}_{p_i} - a_i| \tag{5}$$

where $a_i \in \mathbb{R}^4$ is the ground truth border distance.

E CrowdHuman dataset

CrowdHuman [34]contains 15k training images with common pose annotations. The dataset is featured of high density and large occlusion. Both visible bounding box and the Amodal bounding box are annotated. We use the Amodal bounding box annotation in our experiments to align with MOT [28].

F Pretraining experiments

For pretraining on CrowdHuman [34], we use input resolution 512×512, false positive ratio $\lambda_{fp} = 0.1$, false negative ratio $\lambda_{fn} = 0.4$, random scaling ratio 0.05, and random translation ratio 0.05. The training follows Section.4.4 of the main paper. As shown in Table 9, the model trained on CrowdHuman achieves a decent 52.2 MOTA in MOT dataset, without seeing any MOT data.

Without CrowdHuman [34] pretraining, our performance drops to 60.7% MOTA on the validation set. Pretraining help improve detection quality by decreasing the false negatives. Note that most entries on MOT challenges use external data for pretraining, and some of them use private data [52]. For reference, we also show our public detection results without pretraining in Table 9, last row. This model corresponds to the entry we submitted to MOT17 public detection challenge.

G Additional experiments on KITTI

In Table 10, we show results of the same additional experiments (Section. 5.5 of the main paper) on KITTI dataset [12]. The conclusions are the same as on MOT [28]. Training on static images now performs slightly worse than training on video, mostly due to that KITTI has larger inter-frame motion than MOT. Training without random heatmap noise is much worse than the full model, with a high false-negative rate. And using the Hungarian algorithm works the same as using a greedy matching. Our model without nuScenes [3] achieves 84.5% MOTA on the validation set, this is on-par with other state-of-the-art trackers on KITTI [14, 35, 46] with a heavy detector [30].

	MOTA ↑	IDF1 ↑	MT ↑	ML ↓	FP ↓	FN ↓	IDSW ↓
Ours	66.1	64.2	41.3	21.2	4.5%	28.4%	1.0%
only CrowdH.	52.2	53.8	33.6	25.1	6.7%	39.7%	1.4%
scratch	60.7	62.8	33.0	22.4	4.0%	34.2%	1.0%
scratch-Pub.	57.4	59.6	31.1	27.1	2.1%	39.6%	1.0%

Table 9: Additional experiments on the MOT17 validation set. From top to bottom: our full model, the model trained only on CrowdHuman dataset, our model trained from scratch, and the public detection mode of our model trained from scratch.

	MOTA ↑	MOTP ↑	MT ↑	ML ↓	FP ↓	FN ↓	IDSW ↓
Ours	88.7	86.7	90.3	2.1	5.4%	5.8%	0.1%
Static image	86.8	86.5	88.5	2.2	4.8%	7.9%	0.4%
w.o. noisy hm	80.1	85.3	76.2	7.6	3.8%	16.1%	0.1%
Hungarian	88.7	86.7	90.3	2.1	5.4%	5.8%	0.1%
scratch	84.5	83.2	83.4	2.8	5.7%	9.6%	0.3%

Table 10: Additional experiments on the KITTI validation set. From top to bottom: our full model, the public-detection configuration of our model, our model trained on static images only, our model trained without simulating heatmap noise, our model with the Hungarian algorithm used for matching, and our model trained from scratch.

θ	τ	MOTA ↑	IDF1 ↑	MT ↑	ML ↓	FP ↓	FN ↓	IDSW ↓
0.4	0.4	62.6	64.9	44.0	18.9	10.3%	26.4%	0.7%
0.4	0.6	65.5	63.2	38.6	22.4	2.5%	30.5%	1.5%
0.4	0.5	66.1	64.2	41.3	21.2	4.5%	28.4%	1.0%
0.3	0.5	66.2	64.3	43.1	19.2	5.7%	26.9%	1.2%
0.5	0.5	65.2	62.1	39.8	23.0	3.7%	30.2%	0.9%

Table 11: Experiments with different output thresholds (θ) and rendering thresholds (τ) on the MOT [28] validation set. We search θ and τ locally in a step of 0.1.

H Output and rendering threshold

As the tracking evaluation metric (MOTA) does not consider the confidence of predictions, picking an output threshold is essential in all tracking algorithms (see discussion in AB3D [46]). In our case, we also need a threshold to render predictions to the prior heatmap. We search the optimal thresholds on MOT [28] in Table 11. Basically, increasing both thresholds results in fewer outputs, thus increases the false negatives while decreases the false positives. We find a good balance at $\theta = 0.4$ and $\tau = 0.5$.