CAMOT: Camera Angle-aware Multi-Object Tracking

Felix Limanta
Tokyo Institute of Technology
Meguro, Tokyo, Japan
felix@ks.c.titech.ac.jp

Kuniaki Uto
Tokyo Institute of Technology
Meguro, Tokyo, Japan
uto@ks.c.titech.ac.jp

Koichi Shinoda
Tokyo Institute of Technology
Meguro, Tokyo, Japan
shinoda@c.titech.ac.jp

Abstract

This paper proposes CAMOT, a simple camera angle estimator for multi-object tracking to tackle two problems: 1) occlusion and 2) inaccurate distance estimation in the depth direction. Under the assumption that multiple objects are located on a flat plane in each video frame, CAMOT estimates the camera angle using object detection. In addition, it gives the depth of each object, enabling pseudo-3D MOT. We evaluated its performance by adding it to various 2D MOT methods on the MOT17 and MOT20 datasets and confirmed its effectiveness. Applying CAMOT to ByteTrack, we obtained 63.8% HOTA, 80.6% MOTA, and 78.5% IDF1 in MOT17, which are state-of-the-art results. Its computational cost is significantly lower than the existing deep-learning-based depth estimators for tracking.

1. Introduction

Multi-object tracking (MOT) [3, 4, 11, 18, 61, 62] is a task to detect and track objects in a video across space and time while maintaining consistent identities. It is utilized in several applications, such as autonomous driving and video surveillance. Its standard paradigm consists of two stages: 1) object detection [19, 21, 29, 40, 41, 65], wherein it detects individual objects in each frame, and 2) association [7, 62], wherein it associates detection results over time to form a track for each object. In this paper, we focus on the application of MOT to surveillance.

MOT faces several challenges in real-world scenarios. One significant problem is that the target object is often occluded by other objects, resulting in detection failure. Another problem is that the distance between two objects cannot be precisely estimated when they are aligned in the depth direction. This may cause incorrect object associations between different frames.

These two problems can be addressed if we know the depth of each object. To this end, Khurana *et al.* [24] plugged a depth estimator based on deep learning into the MOT framework. Although it somewhat solves the occlusion problem, the imprecise distance problem still needs to be solved. In addition, the depth estimator may require significant additional computational costs.

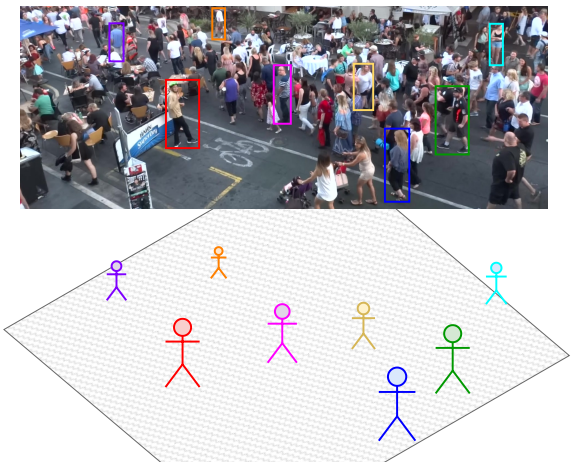

Figure 1. **Illustration on the idea of CAMOT.** Under the assumption that multiple objects are located on a flat plane, the camera angle is estimated using object detection. The scale of each bounding box indicates the depth of each object, whereas the distribution of the bounding boxes informs us of the camera angle.

In this paper, we propose CAMOT (**C**amera **A**ngle-aware **M**ulti-**O**bject **T**racking), a simple camera angle estimator for MOT, to solve these problems. Assuming that multiple objects are located on a flat plane in each video frame, it estimates the camera angle by utilizing object detection. Our method provides the depth of each object and solves the occlusion problem. It additionally measures the distance in the depth direction and associates objects in different frames more accurately. CAMOT is computationally efficient and can be used as a plug-in component in various MOT methods.

We evaluated its performance by adding it to various 2D MOT methods on the MOT17 and MOT20 datasets and confirmed its effectiveness. For example, when applied to ByteTrack, it achieved state-of-the-art results of 63.8% HOTA, 80.6% MOTA, and 78.5% IDF1 in MOT17 [35]. With regards to its computational cost, on a machine with a sin-

gle A100 GPU, it achieved a speed of 24.92 FPS, which is higher than the sub-10 FPS speed of the existing deep-learning-based depth estimators that are used for tracking.

Overall, the main contributions of this work are summarized as follows:

1. We propose a lightweight camera angle estimator that leverages object detection locations.
2. We utilize the camera angle and the depth of each object to associate objects between frames in 2D MOT.
3. We evaluate our proposed method by adding it to various 2D MOT methods.

2. Related Works

2.1. 2D Multi-Object Tracking (MOT)

With the advent of reliable object detection, the standard approach for MOT is "tracking-by-detection," which uses pre-trained detectors and focuses more on data association. Early methods such as SORT [4] and DeepSORT [52] utilize Kalman filters. In contrast, recent methods try novel approaches, such as regressing bounding boxes by frame [3], matching heatmaps in a receptive field [64], or combining detection and reidentification (Re-ID) in a single model [26, 50, 53, 62]. Additionally, the Vision Transformer [16] has also made its way into MOT [10, 46, 54, 59], combining detection and tracking in an end-to-end manner. A notable recent development in MOT is ByteTrack [61], which modifies the simple SORT into a two-pass algorithm that processes low-confidence bounding boxes.

One problem with MOT is that the distance between two objects cannot be precisely estimated when they are aligned in the depth direction. This is because the depth direction is flattened and combined with another direction (usually the up/down direction) when a scene is projected onto a camera. Due to perspective, objects at different depths may appear to have the same distance in the image. However, none of the present methods consider perspective distortion in spatial directions when associating objects.

The most commonly used feature for association is the Intersection-over-Union (IoU) [3, 4, 61], followed by appearance features [52, 62]. Transformer-based MOT [10, 33, 59] attempts to unify detection and tracking and performs the association process as part of the model. All these methods use the distance measured on the 2D image.

A recent alternative to the vanilla IoU is the Distance-IoU (DIoU) [63], which also considers the relative positioning of objects. For two different objects i and j, the DIoU is defined as

$$\mathrm{DIoU} = \mathrm{IoU} - \frac{d_x^2 + d_y^2}{c_x^2 + c_y^2}, \tag{1}$$

where (d_x, d_y) are the horizontal and vertical distances in the image plane between the center points of i and j, and (c_x, c_y) are the width and height of the minimum bounding box that covers the bounding boxes of both i and j. Previous studies have applied DIoU to MOT [28, 45, 57]. We base our own association metric on the DIoU and modify it to incorporate camera angles.

2.2. Occlusion

Occlusion remains a major problem in MOT. A common method to handle occlusions is reidentification (Re-ID), which is used to relink detections before and after occlusions [3, 26, 50, 53, 62, 64]. However, this is a *post-hoc* reasoning on the presence of occluded objects; in an online setting, intelligent agents should reason about occluded objects *before* they re-appear [24]. Furthermore, Re-ID cannot handle longer occlusions (> 3 s) effectively [15] owing to the widening gap between pre- and post-occlusion features.

A possible solution is to lift the tracking process into 3D space. It is much easier to track occluded objects in 3D because trajectories that overlap in 2D are well-separated in 3D space. The most straightforward approach to lifting 2D information to 3D is to use a monocular depth estimator.

2.3. Depth Estimation

Monocular depth estimation obtains a depth map from a single image without additional sensors or modalities. This is an ill-posed problem, as a 2D scene can be projected from infinitely many 3D scenes. Classical methods [36, 42] rely on a thorough *a priori* knowledge of a scene and/or objects in the scene. In contrast, recent deep-learning-based methods can reliably infer depth from a single image without requiring geometric constraints or *a priori* knowledge. The methods range from hourglass networks [27], encoder-decoder structures [5, 58], transformers [1, 25, 39], to diffusion networks [43]. Depth estimation is a precursor to several tasks, including 3D detection and tracking.

Datasets for 3D tasks (*e.g.*, depth estimation, 3D detection, and 3D tracking) are usually collected for tasks such as autonomous driving [9, 22, 47]. As a result, there are currently no available datasets for use cases such as surveillance.

2.4. Depth Estimation for MOT

Plenty of work has been done on monocular 3D object detection and tracking (*i.e.*, generating 3D bounding boxes) with only a single RGB image. Early 3D object detection methods [2, 44, 56, 60] work on point clouds taken by additional sensors (*e.g.*, LiDAR, time-of-flight camera, stereo camera) or generated by monocular depth estimators. In contrast, recent methods [8, 30, 49] generate 3D bounding box proposals from a single 2D image. Several 2D MOT methods [3, 4, 52, 61] can also perform 3D MOT with only slight modifications. Additionally, dedicated 3D trackers [12, 51] have also been proposed.

There also exist studies that use depth information for 2D MOT. However, unlike 3D MOT, these studies still use 2D bounding boxes as input and output; however, internal processing, such as association and trajectory management, is performed in 3D. The first study to add 3D reasoning for 2D MOT was Khurana *et al.* [24], which utilized monocular depth estimation [27] to generate a depth map and augment SORT. By modifying the Kalman filter to track in 3D, the tracker can filter false positives for forecasts that are supposed to be occluded, thus allowing for better occlusion handling. Another work, Quo Vadis [15], transforms the entire scene into a birds-eye view using homography transformation and then adds trajectory prediction. Depth estimators are computationally expensive [58]; Quo Vadis [15] adds other computationally expensive components besides the depth estimator, making the system too costly to run in any practical scenario. In comparison, our proposed estimator requires much less computational cost because it uses readily available bounding boxes.

2.5. Camera Pose Estimation Methods

Traditional camera pose estimation methods, *e.g.*, RANSAC [20] and PnP [31], require handcrafted features to match between images. In contrast, deep-learning-based monocular camera pose estimation methods [6, 23] (which also produce camera angles) have produced good results with only a single image. However, most camera pose estimation methods require a video from a dynamic moving camera or require a well-defined structure as a reference. Unfortunately, many MOT scenes are captured with a static camera and contain large crowds, which may obscure the reference structure.

Our proposed method estimates camera angles with object detection output without requiring a reference structure or a dynamic camera. In addition, it can estimate camera-relative 3D points of every object without any additional training.

3. Camera Angle Estimation

3.1. Outline

Here, we describe our method for estimating the camera elevation angle θ and the set of object 3D coordinates $\mathbb{P}$. CAMOT simultaneously estimates the angle and object depths by regressing a common plane for all object detections.

Object detections, among other things, inform us where objects are distributed on an image, whereas their distributions inform us of the camera angle. For example, an image taken from a ground-level angle will have its objects concentrated in a horizontal line, whereas an image with a higher angle will have its objects distributed more evenly. We can use object detections to estimate the depth of an object, which can then be used to estimate the camera angle.

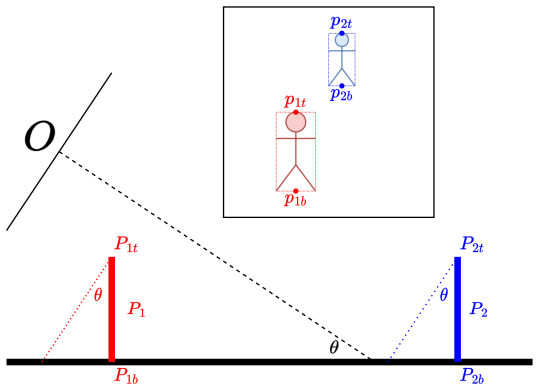

Figure 2. 2D planar side view of the system. **Black** parts show the part of the system shared by all objects, whereas **blue** and **red** parts show different objects.

An outline of our algorithm is as follows:

1. Select bounding boxes to use in the current frame t.
2. While θ^t is not optimal $\left(\varepsilon^{(t,u)} > \tau_\varepsilon\right)$, advance the iteration $u \leftarrow u + 1$ as outlined below:
   (a) Set a $\theta^{(t,u)}$ value for the current iteration.
   (b) Estimate the 3D object points $P_i^{(t,u)}$ using $\theta^{(t,u)}$.
   (c) Regress a plane with the normal vector $\mathbf{n}^{(t,u)}$ from $P_i^{(t,u)}$ and calculate the plane angle $\theta_{\mathbf{n}}^{(t,u)}$.
   (d) Evaluate the angle estimation process error $\varepsilon^{(t,u)}$ for this iteration.
3. Apply angle smoothing for θ^t.
4. Use the optimal θ^t value to calculate P_i^t for *all* objects in the current frame.

3.2. Assumptions and Problem Formulation

For our intended use case (surveillance, crowd analysis, etc.), we limit the scenario to tracking the movements of a crowd (of humans) in a public space. We set the following assumptions regarding the scenario:

1. The camera parameters (*e.g.*, θ) are unknown, except for the focal length f.
2. At least **three** objects are detected in each frame of the video.
3. An object is assumed to be in contact with the ground, *i.e.*, the bottom edges of all objects lie on a common plane (ground plane).
4. The change of camera angles is smooth over time.

We model a human object as a cylinder in 3D space with a centroid $P_i = (X_i, Y_i, Z_i)$, height H, and an aspect ratio A. We model the height H as constant, but the aspect ratio A can vary. We use pinhole camera optics with a focal length f.

Under the pinhole camera model, we formulate the problem as finding the optimal θ value that minimizes the regression error ε for the best-fit plane between all detected objects. Figure 2 illustrates the problem, where an object i is represented as a line segment $\overline{P_{it}P_{ib}}$ passing through P_i, with P_{it}, P_i, and P_{ib} as the top, middle, and bottom points of an object, respectively.

3.3. Bounding Box Selection

Not all bounding boxes produced by the object detection process can be used for the plane estimation process. Let i be an index for all detected objects in the current frame. We first need to select n_{plane} objects, where n_{plane} is the target number of objects to use in the plane estimation process.

We first filter out objects whose corresponding bounding boxes clip the edge of the frame. We then divide the image into n_{plane} regions width-wise. For each region, we add to $\mathbb{I}_{\text{plane}}$ at most one object with the highest detection confidence whose centroid lies in that region. It is possible for a region to be empty (*i.e.*, does not contain any detection). In that case, $|\mathbb{I}_{\text{plane}}|$ may be less than n_{plane}.

3.4. Initial Elevation Angle Setting

We define the camera elevation angle θ as the angle between the camera principal axis (*i.e.*, the z-axis) and its projection on the ground plane. Given the set of 3D points $\mathbb{P} = \{P_i\}$, it is trivial to obtain θ (see Section 3.6). However, as both θ and $\mathbb{P}$ are unknown at the beginning, we first assume an initial θ value and iteratively obtain a final θ value through optimization (see Section 3.7).

In the first frame, θ is initialized according to the parameter θ^0. In later frames, θ is initialized as the previous frame's value.

3.5. Depth Estimation Using Detection Results

This process is first performed in each optimization iteration over the selected objects $\mathbb{I}_{\text{plane}}$ to obtain the set of selected points $\mathbb{P}_{\text{plane}} = \{P_i\}$. After the optimal value for θ is obtained, one last pass is performed over the set of all objects $\mathbb{I}$ to obtain the set of all 3D points $\mathbb{P}$.

Using the angle θ, we can find the distance d_i from the origin point to P_i, P_{it}, and P_{ib}. We use the top and bottom points for calculation since they are the nearest and furthest points an object has from the camera origin point, respectively.

We first convert 2D coordinates into 3D rays originating from the camera focal point, defined as

$$r_i = \kappa_c^{-1} \begin{bmatrix} x_i & y_i & 1 \end{bmatrix}^T, \tag{2}$$

where the ray $r_i = (a_i, b_i, 1)$ is the ray passing through the pinhole camera origin point, the relevant point in the image plane, and the actual point in the 3D camera coordinate, and κ_c^{-1} is the inverse of the camera intrinsic matrix. For

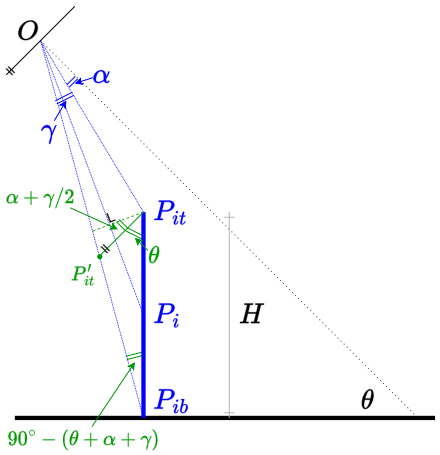

Figure 3. 2D planar side view for one object. **Black** parts show part of the system shared by all objects, while **blue** parts show components unique for the object i. **Green** parts show derived points, angles, etc., for calculation.

every object i, we obtain the rays to the top point r_{it} and the bottom point r_{ib} by using the top-middle and bottom-middle 2D positions $(x_i, y_{it}), (x_i, y_{ib})$ of a bounding box, respectively, as illustrated in Fig. 2.

We then define the following: for each object i, the angle α_i is defined as the angle between the principal axis and the r_{it}, whereas the angle γ_i is defined as the angle between r_{it} and r_{ib}.

In Fig. 3, we can use the angle properties in the triangles formed by the object and in the rays to calculate the distances from the origin point to the object. Using the triangle $\triangle OP_{it}P_i$, the distance to the head point d_{it} can be obtained as follows.

$$d_{it} = \frac{H \cos\left(\theta + \alpha_i + \frac{\gamma_i}{2}\right)}{2 \sin\frac{\gamma_i}{2}} \tag{3}$$

Then, using $\triangle OP_{it}P_{ib}$, the distance to the bottom point d_{ib} can be obtained as follows.

$$d_{ib} = \frac{2d_{it} \sin\left(\frac{\gamma_i}{2}\right) \sin\left(\theta + \alpha_i + \frac{\gamma_i}{2}\right)}{\cos\left(\theta + \alpha_i + \gamma_i\right)} + d_{it} \tag{4}$$

Afterwards, using the rays (r_{it}, r_{ib}), the centroid point P_i is calculated following Eq. 5.

$$P_i = \frac{r_{it}d_{it}\cos\alpha_i + r_{ib}d_{ib}\cos\left(\alpha_i + \gamma_i\right)}{2} \tag{5}$$

Applying this to the selected objects $\mathbb{I}_{\text{plane}}$, we obtain the set of 3D object points $\mathbb{P}_{\text{plane}}$ to use in the plane estimation process. In the final pass, applying this to all objects $\mathbb{I}$, we obtain the final set of all 3D object points $\mathbb{P}$.

3.6. Plane Estimation

$\mathbb{P}_{\text{plane}}$ is fit to a plane $(\mathbf{n}, P_{\text{plane}})$, with $\mathbf{n}$ being the plane normal unit vector and $P_{\text{plane}} = (0, 0, z_{\text{plane}})$ being the inter-

section of the z-axis and the plane.

Finally, the plane is used to obtain the angle of the plane $\theta_{\mathbf{n}}$. Provided that $\mathbf{n} = (n_x, n_y, n_z)$ is a unit vector, $\theta_{\mathbf{n}}$ is obtained with

$$\theta_{\mathbf{n}} = \arccos(n_z) \tag{6}$$

Note that plane estimation in 3D requires at least three points. If there are too few objects in the frame (*i.e.*, $|\mathbb{P}_{\text{plane}}| < 3$), then we keep the plane from the previous frame (*i.e.*, $(\mathbf{n}, P_{\text{plane}})^t = (\mathbf{n}, P_{\text{plane}})^{t-1}$)

3.7. Error Calculation

We aim to find the elevation angle value θ that minimizes the error ε. We calculate three error terms: 1) the perpendicularity constraint $\varepsilon_{\mathbf{n}}$, in which an object should stand perpendicular to the plane, *i.e.*, $\mathbf{n}$ and the vector $\mathbf{v}_i = P_{it} - P_{ib}$ should be parallel (Eq. 7); 2) the plane angle error ε_θ, which is the normalized angle difference between the assumed θ and the angle of the regressed plane $\theta_{\mathbf{n}}$ in radians (Eq. 8); and 3) the regression error $\varepsilon_{\text{regr}}$, which is the root mean squared error (RMSE) for the distance between P_i and the regressed plane (Eq. 9).

$$\varepsilon_{\mathbf{n}} = \frac{1}{n}\sum_{i=1}^{n}\left[1 - \frac{\mathbf{v}_i \cdot \mathbf{n}}{\|\mathbf{v}_i\|\,\|\mathbf{n}\|}\right] \tag{7}$$

$$\varepsilon_\theta = \frac{2}{\pi}\,|\theta - \theta_{\mathbf{n}}| \tag{8}$$

$$\varepsilon_{\text{regr}} = \sqrt{\frac{1}{n}\sum_{i=1}^{n}\|[\mathbf{n}\cdot(P_i - P_{\text{plane}})]\times\mathbf{n}\|^2} \tag{9}$$

We calculate the final error ε as a weighted sum of the three error terms:

$$\varepsilon = \lambda_{\mathbf{n}}\varepsilon_{\mathbf{n}} + \lambda_\theta\varepsilon_\theta + \lambda_{\text{regr}}\varepsilon_{\text{regr}} \tag{10}$$

We use the Nelder–Mead optimization algorithm [37] with the error ε as the minimization objective when finding the optimal angle. The optimization process stops when ε is smaller than the convergence threshold τ_ε.

The final product is the optimal elevation angle θ.

3.8. Angle Smoothing

To enforce the assumption that depth estimates (and thus angles) are smooth over time, we apply weighted moving average smoothing with a window of $w = \text{fps}/2$.

$$\theta^t = \frac{2}{w(w+1)}\sum_{i=t-w+1}^{t}\left[(i-t+w)\theta^i\right] \tag{11}$$

Using the smoothed angle θ^t for frame t, we recalculate the set of all points $\mathbb{P}^t$ to obtain the final representation for frame t.

The camera angles and object depths produced may not be accurate with the ground truth; however, as long as the angles and depths are consistent (*i.e.*, smooth over time) throughout the video, our method does not require them to be accurate.

4. Angle- and Depth-aware MOT

4.1. Tracking in 3D Camera Coordinates

Many MOT methods [4, 52, 61] employ the Kalman filter to predict the location of objects and reconcile it with the detected object positions. The state space of a Kalman filter used in tracking typically includes the position (x, y), aspect ratio (a), height (h), and their velocities $(\dot{x}, \dot{y}, \dot{a}, \dot{h})$. We add an inverse depth term $1/z$ to this state space, which gives the state vector $\left(x_i, y_i, \frac{1}{z_i}, a_i, \dot{x}_i, \dot{y}_i, \dot{h}_i\right)$ [24]. The values of (x_i, y_i, z_i) are obtained from the 3D centroid point P_i produced in Section 3.5.

To account for variations in depth, following [24], we scale the Gaussian noise by the inverse depth, resulting in a constant velocity model, as shown in Eq. 12. This leads to smoother tracks for objects that are far away.

$$x_t \approx x_{t-1} + \dot{x}_{t-1} + f\frac{\epsilon_x}{z_{t-1}} \tag{12}$$

To account for camera motion, following [3, 24], we apply camera motion compensation (CMC) by aligning frames via image registration using the Enhanced Correlation Coefficient (ECC) maximization introduced in [17].

4.2. Track Association

To account for the camera elevation angle, we modify the similarity matrix $K = [k_{ij}]$ based on the Distance-IoU [63]. From the original formula in Eq 1, we add the camera angle factor ϕ to the y-axis terms (d_y and c_y). The modified similarity measure is defined as

$$k'_{ij} = k_{ij} - \frac{d_x^2 + \phi d_y^2}{c_x^2 + \phi c_y^2}, \tag{13}$$

$$\phi = 1 + \cos^2\theta, \tag{14}$$

where i and j are the detection and track, respectively, k_{ij} is the original similarity measure (2D IoU), and k'_{ij} is the modified similarity measure (DIoU with angles).

Low elevation angles cause ϕ to have a large value and k'_{ij} a smaller value, thereby reducing the similarity between two objects if they are positioned above or below each other. This discourages searching in the image's vertical direction, as objects typically do not move vertically when the camera is at ground level. For high elevation angles, where objects have more freedom of movement around the vertical axis, ϕ has less effect and Eq. 13 degrades to the normal DIoU similarity.

The resulting similarity matrix $K' = [k'_{ij}]$ incorporates camera angle information and is used in the object association step.

4.3. Assumptions and Limitations

CAMOT also relies on the assumptions stated in Section 3.2. Particularly, for tracking in 3D coordinates, we assume that the camera focal length f is known (Assumption 1). In practical applications, it is possible to calibrate the camera to satisfy this assumption. However, most videos found on the Internet (*e.g.*, YouTube, etc.), including the videos in MOT evaluation datasets, do not come with camera intrinsics. Thus, we tune the value of f on the training set and select a single f value that best fits our dataset. We observe that the value of f can be generalized sufficiently for typical video sequences.

Another important assumption is that the change of camera angles (and thus depth estimates) is smooth over time (Assumption 4). Once again, our method works even though the values are not accurate, as long as they are consistent.

In addition, CAMOT benefits when more objects are detected in the current frame, as the plane estimation would be more stable. However, as mentioned in Section 3.6, CAMOT only works if there is a minimum of three objects at any time in the frame.

Although these assumptions and limitations may not always hold in real-world scenarios, our empirical results show that our method is applicable to different scenarios.

5. Experiments

5.1. Experiment Setup

Datasets. We evaluate our method on the MOTChallenge [14, 35] (*i.e.*, MOT17 and MOT20) datasets. As standard protocols, CLEAR MOT Metrics [35] and HOTA [32] are used for evaluation.

Implementation details. We implemented our proposed method in PyTorch [38] and performed all experiments on a system with 8 NVIDIA Tesla A100 GPUs. We used ByteTrack [61] as a baseline and built our method on its top. Most hyperparameters that are used for tracking are kept the same, with the detection thresholds $\tau_{\text{high}}, \tau_{\text{low}}$ kept as 0.6 and 0.2, respectively. Lost tracklets were kept for 30 frames before being discarded.

For object detection, we used YOLOX-x [21] pre-trained in COCO. The model was trained on a mix of MOT17, MOT20, CrowdHuman, Cityperson, and ETHZ for 80 epochs with a batch size of 48. We used SGD as an optimizer with a weight decay of 5×10^{-4} and a momentum of 0.9. The initial learning rate is 10^{-3} with a 1 epoch warm-up and cosine annealing schedule.

For angle estimation, we excluded all objects with bounding boxes partially outside the frame. We set $f = 50\text{mm}$ and $n_{\text{plane}} = 40$. For objects clipped on the top or bottom edges, we used the 3D object height H defined in 3.2 to extrapolate the clipped side in relation to the visible side. For objects clipped on the left or right edges, we extrapolated the clipped bounding box points using the average aspect ratio. We empirically set the initial input angle θ^0 to $0°$, the error weights $[\lambda_{\mathbf{n}}, \lambda_\theta, \lambda_{\text{regr}}]$ to $[0.6, 0.3, 0.1]$, and the convergence threshold τ_ε to 10^{-4}.

Following [21], FPS was measured using FP16 precision [34] and a batch size of 1 on a single GPU. We use a machine running an AMD EPYC 7702 1.5GHz with 256GiB RAM and one NVIDIA A100 GPU.

5.2. Results

Benchmarks. Table 1 and Table 2 present a comparison of our tracker with the other mainstream MOT methods on the test sets of MOT17 [35] and MOT20 [14], respectively. Since detection quality significantly affects overall tracking performance, for a fair comparison, the methods in the lower block use the same detections generated by YOLOX [21], with YOLOX weights for the MOT17 and MOT20 datasets provided by ByteTrack [61] and OC-SORT [11], respectively. We also list the methods in the top block for reference, which may use better or worse detections than ours.

Our method outperforms all previous approaches in HOTA, MOTA, and IDF1, while being slightly inferior on FP, FN, and IDSw. On MOT17, our method exhibits the least identity switch (IDSw) errors compared with other methods using the same detection, whereas on MOT20, we narrowly come second. Our method improves on the original ByteTrack [61] baseline and generally outperforms all other methods.

Compared to the baseline (ByteTrack [61]), we have obtained less #FP and #IDSw, but introduced more #FN. The 3D tracking approach we employ results in better separation of trajectories, leading to a smaller amount of viable detection-track pair candidates.

Inference speed. We also demonstrate that our method can operate in real time without significantly reducing inference speed. As shown in Table 3, although incorporating angle and 3D point estimation do have a slight impact on speed, it is not significant. The benefits of improved performance more than offset any modest decrease in speed.

5.3. Ablation Studies

Component analysis. Table 4 shows the results of the ablation studies conducted using the MOT17 dataset. We test for four variables: whether angle estimation is performed on the entire video or just on the first frame (Varθ); whether the depth forecast is used in the Kalman Filter (DF); whether the angle-aware association is performed (AA); and whether camera motion compensation (CMC) is used. The results show that each component improves

Method	HOTA ↑	MOTA ↑	IDF1 ↑	FP (10^4) ↓	FN (10^4) ↓	IDSw ↓
FairMOT [62]	59.3	73.7	72.3	2.75	11.7	3303
GRTU [48]	62.0	74.9	75.0	3.20	10.8	**1812**
MOTR [59]	57.2	71.9	68.4	2.11	13.6	2115
TransMOT [13]	61.7	76.7	75.1	3.62	9.32	2346
MeMOT [10]	56.9	72.5	69.0	2.72	11.5	2724
UniCorn [55]	61.7	77.2	75.5	5.01	**7.33**	5379
ByteTrack [61]	63.1	80.3	77.3	2.55	8.37	2196
OC-SORT [11]	63.2	78.0	77.5	**1.51**	10.8	1950
CAMOT (ours)	**63.8**	**80.6**	**78.5**	1.85	8.96	1843

Table 1. Results on the MOT17-test dataset with private detections. Methods in the gray block share the same detections.

Method	HOTA ↑	MOTA ↑	IDF1 ↑	FP (10^4) ↓	FN (10^4) ↓	IDSw ↓
FairMOT [62]	54.6	61.8	67.3	10.3	8.89	5243
TransMOT [13]	61.9	77.5	75.2	3.42	**8.08**	1615
MeMOT [10]	54.1	63.7	66.1	4.79	13.8	1938
ByteTrack [61]	61.3	77.8	75.2	2.62	8.76	1223
OC-SORT [11]	62.1	75.5	75.9	**1.80**	10.8	**913**
CAMOT (ours)	**62.8**	**78.2**	**76.1**	2.09	9.13	945

Table 2. Results on the MOT20-test dataset with private detections. Methods in the gray block share the same detections.

Method	FPS ↑
TrackFormer [33]	10.0
ByteTrack [61]	**29.6**
CAMOT (ours)	27.9

Table 3. Comparison of the detection/tracking speeds generated by CAMOT and other existing methods. For CAMOT, all components listed in the ablation study (Table 4) are active here.

Varθ	DF	AA	CMC	MOTA ↑	IDF1 ↑
✓				73.2	74.3
	✓			72.9	72.3
✓	✓			74.4	76.1
✓		✓		73.9	74.2
✓	✓	✓		76.2	79.2
✓	✓	✓	✓	78.4	81.2

Table 4. Ablation study on the MOT17 validation set. Varθ indicates whether angle estimation is performed on the entire video or just on the first frame; DF indicates whether the depth forecast is used in the Kalman Filter; AA indicates whether the angle-aware association is performed; and CMC indicates whether camera motion compensation is used.

tracking performance. In addition, we note that CMC significantly improves the performance, perhaps due to the number of moving camera sequences in the MOT17 dataset.

Per-sequence analysis. To test the effect of camera angles on tracking performance, based on the qualitative camera angle in the first frame, we divided the MOT17 validation set into low-angle ($\theta \leq 15°$) and high-angle ($\theta > 15°$) sequences. The results are shown in Table 5.

In lower-angle scenarios, both tracking in 3D camera coordinates and the angle-aware association significantly impact the tracking performance. By operating in 3D camera coordinates, the tracker better understands object trajectories, mitigating identity switches and improving accuracy, particularly in occlusion cases. The angle-aware association further enhances the tracking process by discouraging unlikely trajectory associations based on the current camera angle.

The angle-aware association module becomes less influential in higher-angle scenarios than in lower-angle cases. However, tracking in 3D camera coordinates remains highly effective in higher-angle scenarios. The elevated camera angle reduces occlusion and perspective distortion caused by objects in the foreground, providing a wider field of view and improved visibility.

Using depth estimators for tracking. We also tried replacing our proposed bounding-box-based depth estimation method with several off-the-shelf monocular depth estimators to evaluate its performance as a tracking component.

Table 6 presents the evaluation results of the modified trackers on the MOT17 validation set. Our proposed depth estimation algorithm outperforms early depth estimators but falls short of more state-of-the-art methods. Early depth estimators often group crowds as a homogeneous blob,

Sequence	MOTA ↑	IDF1 ↑
MOT17-02	46.9 (+4.1)	58.2 (+3.2)
MOT17-05	78.1 (+1.4)	78.3 (+2.1)
MOT17-09	82.8 (+2.2)	79.8 (+2.0)
MOT17-10	68.3 (−1.7)	69.7 (+0.9)
MOT17-11	70.4 (+0.9)	72.8 (+1.0)
MOT17-04	89.7 (+4.8)	92.2 (+2.1)
MOT17-13	78.9 (+1.7)	83.2 (+1.4)
Overall	78.4 (+2.2)	81.2 (+2.4)

Table 5. Per-sequence analysis on the MOT17 validation set. Sequences in the white block are low-angle ($\theta \leq 15°$), whereas those in the gray block are high-angle ($\theta > 15°$).

Method	MOTA ↑	IDF1 ↑	FPS ↑
DPT [39]	74.4	76.2	4.75
GLPN [25]	76.3	77.2	6.24
DepthFormer [1]	78.2	80.3	8.44
NewCRFs [58]	79.1	81.6	7.12
ZoeDepth [5]	**80.2**	**82.3**	8.24
Baseline	78.4	81.2	**24.92**

Table 6. Results of replacing depth estimation with existing monocular depth estimators on the MOT17 validation set. "Baseline" refers to our bounding-box-based method.

whereas later depth estimators exhibit some ability to handle such scenarios. Despite relying solely on bounding box location information, our method remains competitive as a tracking component and offers significantly faster processing speeds.

We hypothesize that the current limitations of monocular depth estimators in handling crowded scenes arise from the training data's incompatibility. These depth estimators are typically trained on datasets designed for autonomous driving, primarily consisting of ground-level camera views capturing vehicles and pedestrians. Consequently, they are less suited to handle high-angle scenarios characterized by densely crowded pedestrians effectively. Utilizing a depth estimator trained on a more suitable dataset with high-angle crowds and proper 3D labeling would likely improve performance.

Application on various MOT methods. To evaluate the versatility of our angle estimation method, we applied it to several state-of-the-art MOT methods, including JDE [50], FairMOT [62], CenterTrack [64], and OC-SORT [11] We used the object detections produced by each tracker and applied our angle estimation method to estimate camera-relative 3D coordinates in the Kalman Filter update step. We also performed angle-aware association, similar to our method, while keeping other settings, such as training datasets and hyperparameters, the same.

Method	MOTA ↑	IDF1 ↑
JDE [50]	60.6 (+0.6)	65.1 (+1.5)
FairMOT [62]	70.3 (+1.2)	73.2 (+0.4)
CenterTrack [64]	67.4 (+1.3)	67.3 (+3.1)
OC-SORT [11]	78.8 (+0.8)	78.1 (+0.6)

Table 7. Results of adding angle estimation and 3D association to other state-of-the-art trackers on the MOT17 validation set.

Table 7 presents the evaluation results of the modified trackers on the MOT17 validation set. The results show that adding angle estimation and 3D modeling improved the tracking performance of each method. These results demonstrate the potential of our method to be integrated with existing trackers.

6. Conclusion

This paper introduces CAMOT, an angle estimator for MOT. By estimating the camera angle, the tracker employs a heuristic to adapt the tracking behavior against the perspective distortion of how objects move relative to the camera. In addition, the calculated object depths also enable pseudo-3D MOT. Applying CAMOT to other 2D MOT trackers, the evaluation results on the MOT17 and MOT20 datasets demonstrate how CAMOT provides performance gains over existing methods and achieves state-of-the-art results. CAMOT is also more computationally efficient than deep-learning-based monocular depth estimators that are used for tracking.

Currently, CAMOT uses only a single frame as input when estimating the camera angle. Our future work will focus on using multiple frames to estimate the camera angle for improved stability. We are also interested in applying CAMOT to general depth estimation problems, where we can safely assume that the room geometry and the sizes of objects are fixed.

References

[1] Ashutosh Agarwal and Chetan Arora. Depthformer: Multi-scale vision transformer for monocular depth estimation with global local information fusion. In *ICIP*. IEEE, 2022. 2, 8

[2] Waleed Ali, Sherif Abdelkarim, Mahmoud Zidan, Mohamed Zahran, and Ahmad El Sallab. YOLO3D: end-to-end real-time 3d oriented object bounding box detection from lidar point cloud. In *ECCV workshops*. Springer, 2018. 2

[3] Philipp Bergmann, Tim Meinhardt, and Laura Leal-Taixé. Tracking without bells and whistles. In *ICCV*. IEEE, 2019. 1, 2, 5

[4] Alex Bewley, ZongYuan Ge, Lionel Ott, Fabio Tozeto Ramos, and Ben Upcroft. Simple online and realtime tracking. In *ICIP*. IEEE, 2016. 1, 2, 5

[5] Shariq Farooq Bhat, Reiner Birkl, Diana Wofk, Peter Wonka, and Matthias Müller. Zoedepth: Zero-shot transfer by combining relative and metric depth. *CoRR*, abs/2302.12288, 2023. 2, 8

[6] Samarth Brahmbhatt, Jinwei Gu, Kihwan Kim, James Hays, and Jan Kautz. Geometry-aware learning of maps for camera localization. In *CVPR*. IEEE, 2018. 3

[7] Guillem Brasó and Laura Leal-Taixé. Learning a neural solver for multiple object tracking. In *CVPR*. IEEE, 2020. 1

[8] Garrick Brazil and Xiaoming Liu. M3D-RPN: monocular 3d region proposal network for object detection. In *ICCV*. IEEE, 2019. 2

[9] Holger Caesar, Varun Bankiti, Alex H. Lang, Sourabh Vora, Venice Erin Liong, Qiang Xu, Anush Krishnan, Yu Pan, Giancarlo Baldan, and Oscar Beijbom. nuscenes: A multimodal dataset for autonomous driving. In *CVPR*. IEEE, 2020. 2

[10] Jiarui Cai, Mingze Xu, Wei Li, Yuanjun Xiong, Wei Xia, Zhuowen Tu, and Stefano Soatto. Memot: Multi-object tracking with memory. In *CVPR*. IEEE, 2022. 2, 7

[11] Jinkun Cao, Xinshuo Weng, Rawal Khirodkar, Jiangmiao Pang, and Kris Kitani. Observation-centric SORT: rethinking SORT for robust multi-object tracking. *CoRR*, abs/2203.14360, 2022. 1, 6, 7, 8

[12] Hsu-Kuang Chiu, Jie Li, Rares Ambrus, and Jeannette Bohg. Probabilistic 3d multi-modal, multi-object tracking for autonomous driving. In *ICRA*. IEEE, 2021. 2

[13] Peng Chu, Jiang Wang, Quanzeng You, Haibin Ling, and Zicheng Liu. Transmot: Spatial-temporal graph transformer for multiple object tracking. In *WACV*. IEEE, 2023. 7

[14] Patrick Dendorfer, Hamid Rezatofighi, Anton Milan, Javen Shi, Daniel Cremers, Ian D. Reid, Stefan Roth, Konrad Schindler, and Laura Leal-Taixé. MOT20: A benchmark for multi object tracking in crowded scenes. *CoRR*, abs/2003.09003, 2020. 6

[15] Patrick Dendorfer, Vladimir Yugay, Aljosa Osep, and Laura Leal-Taixé. Quo vadis: Is trajectory forecasting the key towards long-term multi-object tracking? In *NeurIPS*, 2022. 2, 3

[16] Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, Jakob Uszkoreit, and Neil Houlsby. An image is worth 16x16 words: Transformers for image recognition at scale. In *ICLR*, 2021. 2

[17] Georgios D. Evangelidis and Emmanouil Z. Psarakis. Parametric image alignment using enhanced correlation coefficient maximization. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 30(10):1858–1865, 2008. 5

[18] Christoph Feichtenhofer, Axel Pinz, and Andrew Zisserman. Detect to track and track to detect. In *ICCV*. IEEE, 2017. 1

[19] Pedro F. Felzenszwalb, Ross B. Girshick, David A. McAllester, and Deva Ramanan. Object detection with discriminatively trained part-based models. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 32(9):1627–1645, 2010. 1

[20] Martin A. Fischler and Robert C. Bolles. Random sample consensus: A paradigm for model fitting with applications to image analysis and automated cartography. *Communications of the ACM*, 24(6):381–395, 1981. 3

[21] Zheng Ge, Songtao Liu, Feng Wang, Zeming Li, and Jian Sun. YOLOX: exceeding YOLO series in 2021. *CoRR*, abs/2107.08430, 2021. 1, 6

[22] Andreas Geiger, Philip Lenz, and Raquel Urtasun. Are we ready for autonomous driving? the kitti vision benchmark suite. In *CVPR*. IEEE, 2012. 2

[23] Alex Kendall, Matthew Grimes, and Roberto Cipolla. Posenet: A convolutional network for real-time 6-dof camera relocalization. In *ICCV*. IEEE, 2015. 3

[24] Tarasha Khurana, Achal Dave, and Deva Ramanan. Detecting invisible people. In *ICCV*. IEEE, 2021. 1, 2, 3, 5

[25] Doyeon Kim, Woonghyun Ga, Pyunghwan Ahn, Donggyu Joo, Sewhan Chun, and Junmo Kim. Global-local path networks for monocular depth estimation with vertical cutdepth. *CoRR*, abs/2201.07436, 2022. 2, 8

[26] Wei Li, Yuanjun Xiong, Shuo Yang, Mingze Xu, Yongxin Wang, and Wei Xia. Semi-tcl: Semi-supervised track contrastive representation learning. *CoRR*, abs/2107.02396, 2021. 2

[27] Zhengqi Li and Noah Snavely. Megadepth: Learning single-view depth prediction from internet photos. In *CVPR*. IEEE, 2018. 2, 3

[28] Chao Liang, Zhipeng Zhang, Xue Zhou, Bing Li, Shuyuan Zhu, and Weiming Hu. Rethinking the competition between detection and reid in multiobject tracking. *IEEE Transactions on Image Processing*, 31:3182–3196, 2022. 2

[29] Wei Liu, Dragomir Anguelov, Dumitru Erhan, Christian Szegedy, Scott E. Reed, Cheng-Yang Fu, and Alexander C. Berg. SSD: single shot multibox detector. In *ECCV*. Springer, 2016. 1

[30] Zechen Liu, Zizhang Wu, and Roland Tóth. SMOKE: single-stage monocular 3d object detection via keypoint estimation. In *CVPR workshops*. IEEE, 2020. 2

[31] Xiao Xin Lu. A review of solutions for perspective-n-point problem in camera pose estimation. *Journal of Physics: Conference Series*, 1087(5):052009, Sep 2018. 3

[32] Jonathon Luiten, Aljosa Osep, Patrick Dendorfer, Philip H. S. Torr, Andreas Geiger, Laura Leal-Taixé, and Bastian Leibe. HOTA: A higher order metric for evaluating multi-object tracking. *International Journal of Computer Vision*, 129(2):548–578, 2021. 6

[33] Tim Meinhardt, Alexander Kirillov, Laura Leal-Taixé, and Christoph Feichtenhofer. Trackformer: Multi-object tracking with transformers. In *CVPR*. IEEE, 2022. 2, 7

[34] Paulius Micikevicius, Sharan Narang, Jonah Alben, Gregory F. Diamos, Erich Elsen, David García, Boris Ginsburg, Michael Houston, Oleksii Kuchaiev, Ganesh Venkatesh, and Hao Wu. Mixed precision training. In *ICLR*. OpenReview.net, 2018. 6

[35] Anton Milan, Laura Leal-Taixé, Ian D. Reid, Stefan Roth, and Konrad Schindler. MOT16: A benchmark for multi-object tracking. *CoRR*, abs/1603.00831, 2016. 1, 6

[36] Takaaki Nagai, Takumi Naruse, Masaaki Ikehara, and Akira Kurematsu. Hmm-based surface reconstruction from single images. In *ICIP*. IEEE, 2002. 2

[37] John A. Nelder and Roger Mead. A simplex method for function minimization. *Computer Journal*, 7:308–313, 1965. 5

[38] Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, Alban Desmaison, Andreas Kopf, Edward Yang, Zachary DeVito, Martin Raison, Alykhan Tejani, Sasank Chilamkurthy, Benoit Steiner, Lu Fang, Junjie Bai, and Soumith Chintala. Pytorch: An imperative style, high-performance deep learning library. In *NeurIPS*, 2019. 6

[39] René Ranftl, Alexey Bochkovskiy, and Vladlen Koltun. Vision transformers for dense prediction. In *ICCV*. IEEE, 2021. 2, 8

[40] Joseph Redmon, Santosh Kumar Divvala, Ross B. Girshick, and Ali Farhadi. You only look once: Unified, real-time object detection. In *CVPR*. IEEE, 2016. 1

[41] Shaoqing Ren, Kaiming He, Ross B. Girshick, and Jian Sun. Faster R-CNN: towards real-time object detection with region proposal networks. In *NIPS*, 2015. 1

[42] Ashutosh Saxena, Sung H. Chung, and Andrew Y. Ng. Learning depth from single monocular images. In *NIPS*, 2005. 2

[43] Saurabh Saxena, Abhishek Kar, Mohammad Norouzi, and David J. Fleet. Monocular depth estimation using diffusion models. *CoRR*, abs/2302.14816, 2023. 2

[44] Jieqi Shi, Peiliang Li, Xiaozhi Chen, and Shaojie Shen. You only label once: 3d box adaptation from point cloud to image via semi-supervised learning. *CoRR*, abs/2211.09302, 2022. 2

[45] Daniel Stadler and Jürgen Beyerer. An improved association pipeline for multi-person tracking. In *CVPR*. IEEE, 2023. 2

[46] Peize Sun, Yi Jiang, Rufeng Zhang, Enze Xie, Jinkun Cao, Xinting Hu, Tao Kong, Zehuan Yuan, Changhu Wang, and Ping Luo. Transtrack: Multiple-object tracking with transformer. *CoRR*, abs/2012.15460, 2020. 2

[47] Pei Sun, Henrik Kretzschmar, Xerxes Dotiwalla, Aurelien Chouard, Vijaysai Patnaik, Paul Tsui, James Guo, Yin Zhou, Yuning Chai, Benjamin Caine, Vijay Vasudevan, Wei Han, Jiquan Ngiam, Hang Zhao, Aleksei Timofeev, Scott Ettinger, Maxim Krivokon, Amy Gao, Aditya Joshi, Yu Zhang, Jonathon Shlens, Zhifeng Chen, and Dragomir Anguelov. Scalability in perception for autonomous driving: Waymo open dataset. In *CVPR*. IEEE, 2020. 2

[48] Shuai Wang, Hao Sheng, Yang Zhang, Yubin Wu, and Zhang Xiong. A general recurrent tracking framework without real data. In *ICCV*. IEEE, 2021. 7

[49] Tai Wang, Xinge Zhu, Jiangmiao Pang, and Dahua Lin. FCOS3D: fully convolutional one-stage monocular 3d object detection. In *ICCV workshops*. IEEE, 2021. 2

[50] Zhongdao Wang, Liang Zheng, Yixuan Liu, Yali Li, and Shengjin Wang. Towards real-time multi-object tracking. In *ECCV*. Springer, 2020. 2, 8

[51] Xinshuo Weng and Kris Kitani. A baseline for 3d multi-object tracking. *CoRR*, abs/1907.03961, 2019. 2

[52] Nicolai Wojke, Alex Bewley, and Dietrich Paulus. Simple online and realtime tracking with a deep association metric. In *ICIP*. IEEE, 2017. 2, 5

[53] Mingze Xu, Chenyou Fan, Yuchen Wang, Michael S. Ryoo, and David J. Crandall. Joint person segmentation and identification in synchronized first- and third-person videos. In *ECCV*. Springer, 2018. 2

[54] Yihong Xu, Yutong Ban, Guillaume Delorme, Chuang Gan, Daniela Rus, and Xavier Alameda-Pineda. Transcenter: Transformers with dense queries for multiple-object tracking. *CoRR*, abs/2103.15145, 2021. 2

[55] Bin Yan, Yi Jiang, Peize Sun, Dong Wang, Zehuan Yuan, Ping Luo, and Huchuan Lu. Towards grand unification of object tracking. In *ECCV*. Springer, 2022. 7

[56] Bo Yang, Jianan Wang, Ronald Clark, Qingyong Hu, Sen Wang, Andrew Markham, and Niki Trigoni. Learning object bounding boxes for 3d instance segmentation on point clouds. In *NeurIPS*, 2019. 2

[57] Di Yuan, Xiu Shu, Nana Fan, Xiaojun Chang, Qiao Liu, and Zhenyu He. Accurate bounding-box regression with distance-iou loss for visual tracking. *Journal of Visual Communication and Image Representation*, 83:103428, 2022. 2

[58] Weihao Yuan, Xiaodong Gu, Zuozhuo Dai, Siyu Zhu, and Ping Tan. Neural window fully-connected crfs for monocular depth estimation. In *CVPR*. IEEE, 2022. 2, 3, 8

[59] Fangao Zeng, Bin Dong, Yuang Zhang, Tiancai Wang, Xiangyu Zhang, and Yichen Wei. MOTR: end-to-end multiple-object tracking with transformer. In *ECCV*. Springer, 2022. 2, 7

[60] Hanwei Zhang, Hideaki Uchiyama, Shintaro Ono, and Hiroshi Kawasaki. MOTSLAM: mot-assisted monocular dynamic SLAM using single-view depth estimation. In *IROS*. IEEE, 2022. 2

[61] Yifu Zhang, Peize Sun, Yi Jiang, Dongdong Yu, Fucheng Weng, Zehuan Yuan, Ping Luo, Wenyu Liu, and Xinggang Wang. Bytetrack: Multi-object tracking by associating every detection box. In *ECCV*. Springer, 2022. 1, 2, 5, 6, 7

[62] Yifu Zhang, Chunyu Wang, Xinggang Wang, Wenjun Zeng, and Wenyu Liu. Fairmot: On the fairness of detection and re-identification in multiple object tracking. *International Journal of Computer Vision*, 129(11):3069–3087, 2021. 1, 2, 7, 8

[63] Zhaohui Zheng, Ping Wang, Wei Liu, Jinze Li, Rongguang Ye, and Dongwei Ren. Distance-iou loss: Faster and better learning for bounding box regression. In *AAAI*, 2020. 2, 5

[64] Xingyi Zhou, Vladlen Koltun, and Philipp Krähenbühl. Tracking objects as points. In *ECCV*. Springer, 2020. 2, 8

[65] Xingyi Zhou, Dequan Wang, and Philipp Krähenbühl. Objects as points. *CoRR*, abs/1904.07850, 2019. 1