Kinematic 3D Object Detection in Monocular Video

Garrick Brazil[1], Gerard Pons-Moll[2], Xiaoming Liu[1], and Bernt Schiele[2]

[1] Michigan State University, Computer Science & Engineering
[2] Max Planck Institute for Informatics, Saarland Informatics Campus
[1] {brazilga, liuxm}@msu.edu, [2] {gpons, schiele}@mpi-inf.mpg.de

Abstract. Perceiving the physical world in 3D is fundamental for self-driving applications. Although temporal motion is an invaluable resource to human vision for detection, tracking, and depth perception, such features have not been thoroughly utilized in modern 3D object detectors. In this work, we propose a novel method for monocular video-based 3D object detection which leverages kinematic motion to extract scene dynamics and improve localization accuracy. We first propose a novel decomposition of object orientation and a self-balancing 3D confidence. We show that both components are critical to enable our kinematic model to work effectively. Collectively, using only a single model, we efficiently leverage 3D kinematics from monocular videos to improve the overall localization precision in 3D object detection while also producing useful by-products of scene dynamics (ego-motion and per-object velocity). We achieve state-of-the-art performance on monocular 3D object detection and the Bird's Eye View tasks within the KITTI self-driving dataset.

Keywords: 3D Object Detection, Monocular, Video, Physics-based

1 Introduction

The detection of foreground objects is among the most critical requirements to facilitate self-driving applications [3, 4, 11]. Recently, 3D object detection has made significant progress [9, 21, 23, 24, 37, 45], even while using only a monocular camera [2, 19, 22, 25, 27, 28, 38, 40]. Such works primarily look at the problem from the perspective of *single* frames, ignoring useful temporal cues and constraints.

Computer vision cherishes inverse problems, *e.g.*, recovering the 3D physical motion of objects from monocular videos. Motion information such as object velocity in the metric space is highly desirable for the path planning of self-driving. However, single image-based 3D object detection can not directly estimate physical motion, without relying on additional tracking modules. Therefore, *video-based 3D object detection* would be a sensible choice to recover such motion information. Furthermore, without modeling the physical motion, image-based 3D object detectors are naturally more likely to suffer from erratic and unnatural changes through time in orientation and localization (as exemplified in Fig. 1(a)).
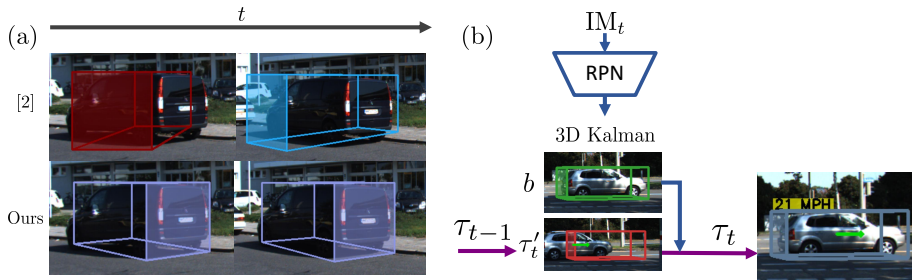
Fig. 1. Single-frame 3D detection [2] often has unstable estimation through time (a), while our video-based method (b) is more robust by leveraging **kinematic motion** via a 3D Kalman Filter to fuse forecasted tracks τ'_t and measurements b into τ_t.

Therefore, we aim to build a novel video-based 3D object detector which is able to provide *accurate* and *smooth* 3D object detection with per-object *velocity*, while also prioritizing a *compact* and *efficient* model overall.

Yet, designing an effective video-based 3D object detector has challenges. Firstly, motion which occurs in real-world scenes can come from a variety of sources such as the camera atop of an autonomous vehicle or robot, and/or from the scene objects themselves — for which most of the safety-critical objects (car, pedestrian, cyclist [14]) are typically *dynamic*. Moreover, using video inherently involves an increase in data consumption which introduces practical challenges for training and/or inference including with memory or redundant processing.

To address such challenges, we propose a novel framework to integrate a 3D Kalman filter into a 3D detection system. We find Kalman is an ideal candidate for three critical reasons: (1) it allows for use of real-world motion models to serve as a strong prior on object dynamics, (2) it is inherently efficient due to its recursive nature and general absence of parameters, (3) the resultant behavior is explainable and provides useful by-products such as the object velocity.

Furthermore, we observe that objects predominantly move in the direction indicated by their orientation. Fortunately, the benefit of Kalman allows us to integrate this real-world constraint into the motion model as a compact scalar velocity. Such a constraint helps maintain the consistency of velocity over time and enables the Kalman motion forecasting and fusion to perform accurately.

However, a model restricted to only move in the direction of its orientation has an obvious flaw — what if the orientation itself is *inaccurate*? We therefore propose a novel reformulation of orientation in favor of accuracy and stability. We find that our orientation improves the 3D localization accuracy by a margin of 2.39% and reduces the orientation error by $\approx 20\%$, which collectively help enable the proposed Kalman to function more effectively.

A notorious challenge of using Kalman comes in the form of *uncertainty*, which is conventionally [41] assumed to be known and static, *e.g.*, from a sensor. However, 3D objects in video are intuitively dependent on more complex factors of image features and cannot necessarily be treated like a sensor measurement. For a better understanding of 3D uncertainty, we propose a 3D self-balancing

confidence loss. We show that our proposed confidence has higher correlation with the 3D localization performance compared to the typical classification probability, which is commonly used in detection [33,35].

To complete the full understanding of the scene motion, we elect to estimate the ego-motion of the capturing camera itself. Hence, we further narrow the work of Kalman to account for only the *object's* motion. Collectively, our proposed framework is able to model important scene dynamics, both ego-motion and per-object velocity, and more precisely detect 3D objects in videos using a stabilized orientation and 3D confidence estimation. We demonstrate that our method achieves state-of-the-art (SOTA) performance on monocular 3D Object Detection and Bird's Eye View (BEV) tasks in the KITTI dataset [14].
In summary, our contributions are as follows:

- We propose a monocular video-based 3D object detector, leveraging realistic motion constraints with an integrated ego-motion and a 3D Kalman filter.

- We propose to reformulate orientation into axis, heading and offset along with a self-balancing 3D localization loss to facilitate the stability necessary for the proposed Kalman filter to perform more effectively.

- Overall, using only a *single model* our framework develops a comprehensive 3D scene understanding including object cuboids, orientation, velocity, object motion, uncertainty, and ego-motion, as detailed in Fig. 1 and 2.

- We achieve a new SOTA performance on monocular 3D object detection and BEV tasks using comprehensive metrics within the KITTI dataset.

2   Related Work

We first provide context of our novelties from the perspective of monocular 3D object detection (Sec. 2.1) with attention to orientation and uncertainty estimation. We next discuss and contrast with video-based object detection (Sec. 2.2).

2.1   Monocular 3D Object Detection

Monocular 3D object detection has made significant progress [2,7,8,19,22,25,27–29,38,39,43]. Early methods such as [8] began by generating 3D object proposals along a ground plane using object priors and estimated point clouds, culminating in an energy minimization approach. [7, 19, 43] utilize additional domains of semantic segmentation, object priors, and estimated depth to improve the localization. Similarly, [27, 40] create a pseudo-LiDAR map using SOTA depth estimator [6, 12, 13], which is respectfully passed into detection subnetworks or LiDAR-based 3D object detection works [18, 31, 32]. In [22, 25, 28, 29, 38] strong 2D detection systems are extended to add cues such as object orientation, then the remaining 3D box parameters are solved via 3D box geometry. [2] extends the region proposal network (RPN) of Faster R-CNN [35] with 3D box parameters.

Orientation Estimation: Prior monocular 3D object detectors estimate orientation via two main techniques. The first method is to classify orientation via a series of discrete bins then regress a relative offset [7, 8, 19, 22, 25, 28, 29, 43].

The bin technique requires a trade-off between the quantity/coverage of the discretized angles and an increase in the number of estimated parameters (bin $\times$). Other methods directly regress the orientation angle using quaternion [38] or Euler [2, 27] angles. Direct regression is comparatively efficient, but may lead to degraded performance and periodicity challenges [46], as exemplified in Fig. 1.

In contrast, we propose a novel orientation decomposition which serves as an intuitive compromise between the bin and direct approaches. We decompose the orientation estimation into three components: axis and heading classification, followed by an angle offset. Thus, our technique increases the parameters by a *static* factor of 2 compared to a bin hyperparameter, while drastically reducing the offset search space for each orientation estimation (discussed in Sec. 3.1).
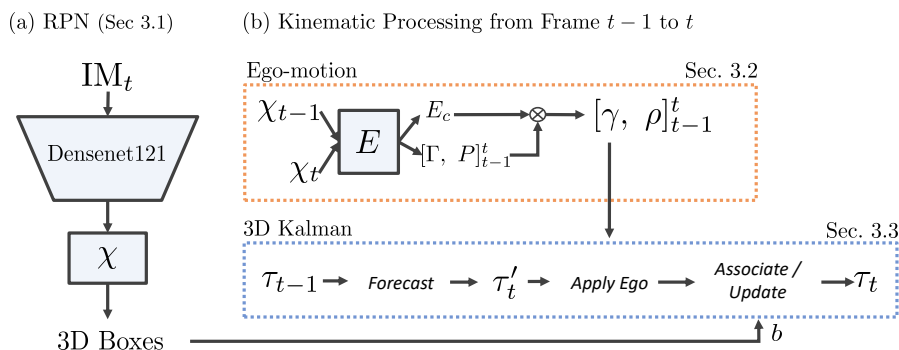
Uncertainty Estimation: Although it is common to utilize the classification score to rate boxes in 2D object detection [5, 33, 35, 44] or explicitly model uncertainty as parametric estimation [20], prior works in monocular 3D object detection realize the need for 3D box uncertainty/confidence [25, 38]. [25] defines confidence using the 3D IoU of a box and ground truth after center alignment, thus capturing the confidence primarily of the 3D *object dimensions*. [38] predicts a confidence by re-mapping the 3D box loss into a probability range, which intuitively represents the confidence of the overall 3D box accuracy.

In contrast, our *self-balancing* confidence loss is generic and self-supervised, with two benefits. (1) It enables estimation of a 3D localization confidence using only the loss values, thus being more general than 3D IoU. (2) It enables the network to naturally re-balance extremely hard 3D boxes and focus on relatively achievable samples. Our ablation (Sec. 4.4) shows the importance of both effects.

2.2   Video-based Object Detection

Video-based object detection [1,26,42,47,48] is generally less studied than single-frame object detection [5, 33–35, 44, 45]. A common trend in video-based detection is to benefit the accuracy-efficiency trade-off via reducing the frame redundancy [1,26,42,47,48]. Such works are applied primarily on domains of ImageNet VID 2015 [36], which contain less ego-motion from the capturing camera than self-driving scenarios [10,14]. As such, the methods are designed to use 2D transformations, which lack the consistency and realism of 3D motion modeling.

In comparison, to our knowledge this is the first work that *utilizes video cues to improve the accuracy and robustness of monocular 3D object detection*. In the domain of 2D/3D object tracking, [15] experiments using Kalman Filters, Particle Filters, and Gaussian Mixture Models, and observe Kalman to be the most effective aggregation method for tracking. An LSTM with depth ordering and IMU camera ego-motion is utilized in [16] to improve the tracking accuracy. In contrast, we explore how to naturally and effectively leverage a 3D Kalman filter to improve the accuracy and robustness of monocular 3D object detection. We propose novel enhancements including estimating ego-motion, orientation, and a 3D confidence, while efficiently using only a single model.

(a) RPN (Sec 3.1)   (b) Kinematic Processing from Frame $t-1$ to t

IM_t

Densenet121

χ

3D Boxes

Ego-motion   Sec. 3.2

χ_{t-1}

E E_c $[\gamma,\ \rho]^t_{t-1}$

χ_t $[\Gamma,\ P]^t_{t-1}$

3D Kalman   Sec. 3.3

$\tau_{t-1} \rightarrow$ *Forecast* $\rightarrow \tau'_t \rightarrow$ *Apply Ego* $\rightarrow$ *Associate / Update* $\rightarrow \tau_t$

b

Fig. 2. Overview. Our framework uses a RPN to first estimate 3D boxes (Sec. 3.1). We forecast previous frame tracks τ_{t-1} into τ'_t using the estimated Kalman velocity. Self-motion is compensated for applying a global ego-motion (Sec. 3.2) to tracks τ'_t. Lastly, we fuse τ'_t with measurements b using a kinematic 3D Kalman filter (Sec. 3.3).
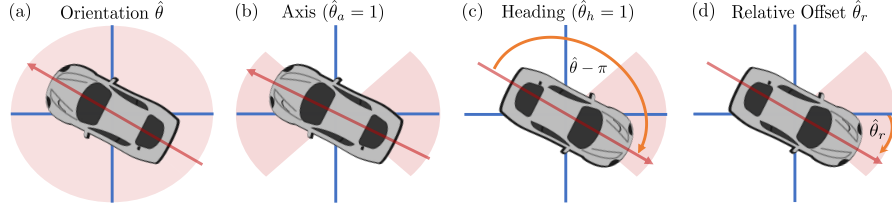
3 Methodology

Our proposed kinematic framework is composed of three primary components: a 3D region proposal network (RPN), ego-motion estimation, and a novel kinematic model to *take advantage of temporal motion in videos*. We first overview the foundations of a 3D RPN. Then we detail our contributions of orientation decomposition and self-balancing 3D confidence, which are integral to the kinematic method. Next we detail ego-motion estimation. Lastly, we present the complete kinematic framework (Fig. 2) which carefully employs a 3D Kalman [41] to model realistic motion using the aforementioned components, ultimately producing a more accurate and comprehensive 3D scene understanding.

3.1 Region Proposal Network

Our measurement model is founded on the 3D RPN [2], enhanced using novel orientation and confidence estimations. The RPN itself acts as a sliding window detector following the typical practices outlined in Faster R-CNN [35] and [2]. Specifically, the RPN consists of a backbone network and a detection head which predicts 3D box outputs relative to a set of predefined anchors.

Anchors: We define our 2D-3D anchor Φ to consist of 2D dimensions $[\Phi_w, \Phi_h]_{2D}$ in pixels, a projected depth-buffer Φ_z in meters, 3D dimensions $[\Phi_w, \Phi_h, \Phi_l]_{3D}$ in meters, and orientations with respect to two major axes $[\Phi_0, \Phi_1]_{3D}$ in radians. The term Φ_z is related to the camera coordinate $[x, y, z]_{3D}$ by the equation $\Phi_z \cdot [u, v, 1]^T_{2D} = \Upsilon \cdot [x, y, z, 1]^T_{3D}$ where $\Upsilon \in \mathbb{R}^{3\times4}$ is a known projection matrix. We compute the anchor values by taking the mean of each parameter after clustering all ground truth objects in 2D, following the process in [2].

3D Box Outputs: Since our network is based on the principles of a RPN [2,35], most of the estimations are defined as a transformation $\mathbf{T}$ relative to an anchor. Let us define n_a as the number of anchors, n_c as the number of object classes,

(a)   Orientation $\hat{\theta}$ (b)   Axis ($\hat{\theta}_a = 1$)   (c)   Heading ($\hat{\theta}_h = 1$)   (d)   Relative Offset $\hat{\theta}_r$

$\hat{\theta} - \pi$

$\hat{\theta}_r$

Fig. 3. Orientation. Our proposed orientation formulation decomposes an object orientation $\hat{\theta}$ (a) into an axis classification $\hat{\theta}_a$ (b), a heading classification $\hat{\theta}_h$ (c), and an offset $\hat{\theta}_r$ (d). Our method disentangles the objectives of axis and heading classification while greatly reducing the offset region (red) by a factor of $\frac{1}{4}$.

and $w \times h$ as the output resolution of the network. The RPN outputs a classification map $\mathbf{C} \in \mathbb{R}^{(n_a \cdot n_c) \times w \times h}$, then 2D transformations $[\mathbf{T}_x, \mathbf{T}_y, \mathbf{T}_w, \mathbf{T}_h]_{2\mathrm{D}}$, 3D transformations $[\mathbf{T}_u, \mathbf{T}_v, \mathbf{T}_z, \mathbf{T}_w, \mathbf{T}_h, \mathbf{T}_l, \mathbf{T}_{\theta_r}]_{3\mathrm{D}}$, axis and heading $[\mathbf{\Theta}_a, \mathbf{\Theta}_h]$, and lastly a 3D self-balancing confidence $\mathbf{\Omega}$. Each output has a size of $\mathbb{R}^{n_a \times w \times h}$. The outputs can be unrolled into $n_b = (n_a \cdot w \cdot h)$ boxes with $(n_c + 14)$-dim, with parameters of c, $[t_x, t_y, t_w, t_h]_{2\mathrm{D}}$, $[t_u, t_v, t_z, t_w, t_h, t_l, t_{\theta_r}]_{3\mathrm{D}}$, $[\theta_a, \theta_h]$, and ω, which relate to the maps by $c \in \mathbf{C}$, $t_{2\mathrm{D}} \in \mathbf{T}_{2\mathrm{D}}$, $t_{3\mathrm{D}} \in \mathbf{T}_{3\mathrm{D}}$, $\theta \in \mathbf{\Theta}$ and $\omega \in \mathbf{\Omega}$. The regression targets for 2D ground truths (GTs) $[\hat{x}, \hat{y}, \hat{w}, \hat{h}]_{2\mathrm{D}}$ are defined as:

$$\hat{t}_{x_{2\mathrm{D}}} = \frac{\hat{x}_{2\mathrm{D}} - i}{\Phi_{w_{2\mathrm{D}}}}, \quad \hat{t}_{y_{2\mathrm{D}}} = \frac{\hat{y}_{2\mathrm{D}} - j}{\Phi_{h_{2\mathrm{D}}}}, \quad \hat{t}_{w_{2\mathrm{D}}} = \log \frac{\hat{w}_{2\mathrm{D}}}{\Phi_{w_{2\mathrm{D}}}}, \quad \hat{t}_{h_{2\mathrm{D}}} = \log \frac{\hat{h}_{2\mathrm{D}}}{\Phi_{h_{2\mathrm{D}}}}, \quad (1)$$

where $(i, j) \in \mathbb{R}^{w \times h}$ represent the pixel coordinates of the corresponding box. Similarly, following the equation of $\hat{z} \cdot [\hat{u}, \hat{v}, 1]_{2\mathrm{D}}^{\mathrm{T}} = \mathbf{\Upsilon} \cdot [x, y, z, 1]_{3\mathrm{D}}^{\mathrm{T}}$, the regression targets for the projected 3D center GTs are defined as:

$$\hat{t}_u = \frac{\hat{u} - i}{\Phi_{w_{2\mathrm{D}}}}, \qquad \hat{t}_v = \frac{\hat{v} - j}{\Phi_{h_{2\mathrm{D}}}}, \qquad \hat{t}_z = \hat{z} - \Phi_z. \qquad (2)$$

Lastly, the regression targets for 3D dimensions GTs $[\hat{w}, \hat{h}, \hat{l}]_{3\mathrm{D}}$ are defined as:

$$\hat{t}_{w_{3\mathrm{D}}} = \log \frac{\hat{w}_{3\mathrm{D}}}{\Phi_{w_{3\mathrm{D}}}}, \qquad \hat{t}_{h_{3\mathrm{D}}} = \log \frac{\hat{h}_{3\mathrm{D}}}{\Phi_{h_{3\mathrm{D}}}}, \qquad \hat{t}_{l_{3\mathrm{D}}} = \log \frac{\hat{l}_{3\mathrm{D}}}{\Phi_{l_{3\mathrm{D}}}}. \qquad (3)$$

The remaining targets for our novel orientation estimation t_{θ_r}, $[\theta_a, \theta_h]$, and 3D self-balancing confidence ω are defined in subsequent sections.

Orientation Estimation: We propose a novel object orientation formulation, with a decomposition of three components: axis, heading, and offset (Fig. 3). Intuitively, the axis estimation θ_a represents the probability an object is oriented towards the vertical axis ($\theta_a = 0$) or the horizontal axis ($\theta_a = 1$), with its label formally defined as: $\hat{\theta}_a = |\sin \hat{\theta}| < |\cos \hat{\theta}|$, where $\hat{\theta}$ is the ground truth object orientation in radians from a bird's eyes view (BEV) with $[-\pi, \pi)$ bounded range. We then compute an orientation $\hat{\theta}_r$ with a restricted range relative to its axis, e.g., $[-\pi, 0)$ when $\hat{\theta}_a = 0$, and $[-\frac{\pi}{2}, \frac{\pi}{2})$ when $\hat{\theta}_a = 1$. We start with $\hat{\theta}_r = \hat{\theta}$ then add or subtract π from $\hat{\theta}_r$ until the desired range is satisfied.

Intuitively, $\hat{\theta}_r$ loses its heading since the true rotation may be $\{\hat{\theta}_r, \ \hat{\theta}_r \pm \pi\}$. We therefore preserve the heading using a separate $\hat{\theta}_h$, which represents the probability of $\hat{\theta}_r$ being rotated by π with its GT target defined as:

$$\hat{\theta}_h = \begin{cases} 0 & \hat{\theta} = \hat{\theta}_r \\ 1 & \text{otherwise.} \end{cases} \tag{4}$$

Lastly, we encode the orientation offset transformation which is relative to the corresponding anchor, axis, and restricted orientation $\hat{\theta}_r$ as: $\hat{t}_{\theta_r} = \hat{\theta}_r - \Phi_{\theta_a}$. The reverse decomposition is $\theta = \Phi_{\theta_a} + \lfloor \theta_h \rceil \cdot \pi + t_{\theta r}$ where $\lfloor \rceil$ denotes round.

In designing our orientation, we first observed that the visual difference between objects at opposite headings of $[\theta, \ \theta \pm \pi]$ is low, especially for far objects. In contrast, classifying the axis of an object is intuitively more clear since the visual features correlate with the aspect ratio. Note that $[\theta_a, \ \theta_h, \ \theta_r]$ disentangle these two objectives. Hence, while the axis is being determined, the heading classifier can focus on subtle clues such as windshields, headlights and shape.

We further note that our 2 binary classifications have the same representational power as 4 bins following [19, 22, 25, 28]. Specifically, bins of $[0, \frac{\pi}{2}, \pi, \frac{3\pi}{2}]$. However, it is more common to use considerably more bins (such as 12 in [19]). An important distinction is that bin-based approaches require the network decide axis and heading simultaneously, whereas our method *disentangles* the orientation into the two distinct and explainable objectives. We provide ablations to compare our decomposition and the bin method using $[2, 4, 10]$ bins in Sec. 4.4.

Self-Balancing Loss: The novel 3D localization confidence ω follows a self-balancing formulation closely coupled to the network loss. We first define the 2D and 3D loss terms which comprise the general RPN loss. We unroll and match all n_b box estimations to their respective ground truths. A box is matched as foreground when sufficient ($\geq k$) 2D intersection over union (IoU) is met, otherwise it is considered background ($\hat{c} = 0$) and all loss terms except for classification are ignored. The 2D box loss is thus defined as:

$$L_{2\text{D}} = -\log(\text{IoU}(b_{2\text{D}}, \ \hat{b}_{2\text{D}}))[\hat{c} \neq 0] + \text{CE}(c, \ \hat{c}), \tag{5}$$

where CE denotes a softmax activation followed by logistic cross-entropy loss over the ground truth class $\hat{c}$, and IoU uses predicted $b_{2\text{D}}$ and ground truth $\hat{b}_{2\text{D}}$. Similarly, the 3D localization loss for only foreground ($\hat{c} \neq 0$) is defined as:

$$L_{3\text{D}} = L_1(t_{3\text{D}}, \ \hat{t}_{3\text{D}}) + \lambda_a \cdot \text{BCE}([\theta_a, \ \theta_h], \ [\hat{\theta}_a, \ \hat{\theta}_h]), \tag{6}$$

where BCE denotes a sigmoid activation followed by binary cross-entropy loss. Next we define the final self-balancing confidence loss with the ω estimation as:

$$L = L_{2\text{D}} + \omega \cdot L_{3\text{D}} + \lambda_L \cdot (1 - \omega), \tag{7}$$

where λ_L is the rolling mean of the n_L most recent $L_{3\text{D}}$ losses per mini-batch. Since ω is predicted per-box via a sigmoid, the network can intuitively balance whether to use the loss of $L_{3\text{D}}$ or incur a proportional penalty of $\lambda_L \cdot (1 - \omega)$. Hence, when the confidence is high ($\omega \approx 1$) we infer that the network is confident in its 3D loss $L_{3\text{D}}$. Conversely, when the confidence is low ($\omega \approx 0$), the network

is uncertain in L_{3D}, thus incurring a flat penalty is preferred. At inference, we fuse the self-balancing confidence with the classification score as $\mu = c \cdot \omega$.

The proposed self-balancing loss has two key benefits. Firstly, it produces a useful 3D localization confidence with inherent correlation to 3D IoU (Sec. 4.4). Secondly, it enables the network to re-balance samples which are exceedingly challenging and re-focus on the more reasonable targets. Such a characteristic can be seen as the inverse of hard-negative mining, which is important while monocular 3D object detection remains highly difficult and unsaturated (Sec. 4.1).

3.2   Ego-motion

A challenge with the dynamics of urban scenes is that not only are most foreground objects in motion, but the capturing camera itself is dynamic. Therefore, for a full understanding of the scene dynamics, we design our model to additionally predict the self-movement of the capturing camera, *e.g.*, ego-motion.

We define ego-motion in the conventional six degrees of freedom: translation $[\gamma_x, \gamma_y, \gamma_z]$ in meters and rotation $[\rho_x, \rho_y, \rho_z]$ in radians. We attach an ego-motion features layer E with ReLU to the concatenation of two temporally adjacent feature maps χ which is the final layer in our backbone with size $\mathbb{R}^{n_h \times w \times h}$ architecture, defined as $\chi_{t-1} \parallel \chi_t$. We then attach predictions for translation $[\Gamma_x, \Gamma_y, \Gamma_z]$ and rotation $[P_x, P_y, P_z]$, which are of size $\mathbb{R}^{w \times h}$. Instead of using a global pooling, we predict a spatial confidence map $E_c \in \mathbb{R}^{w \times h}$ based on E. We then apply softmax over the spatial dimension of E_c such that $\sum E_c = 1$. Hence, the pooling of prediction maps $[\Gamma, \ P]$ into $[\gamma, \ \rho]$ is defined as:

$$\gamma = \sum_{(i,j)} \Gamma(i,j) \cdot E_c(i,j), \qquad \rho = \sum_{(i,j)} P(i,j) \cdot E_c(i,j), \qquad (8)$$

where (i,j) is the coordinate in $\mathbb{R}^{w \times h}$. We show an overview of the motion features E, spatial confidence E_c, and outputs $[\Gamma, \ P] \rightarrow [\gamma, \ \rho]$ within Fig. 2. We use a L_1 loss against GTs $\hat{\gamma}$ and $\hat{\rho}$ defined as $L_{\text{ego}} = L_1(\gamma, \ \hat{\gamma}) + \lambda_r \cdot L_1(\rho, \ \hat{\rho})$.

3.3   Kinematics

In order to leverage temporal motion in video, we elect to integrate our RPN and ego-motion into a novel kinematic model. We adopt a 3D Kalman [41] due to its notable efficiency, effectiveness, and interpretability. We next detail our proposed motion model, the procedure for forecasting, association, and update.

Motion Model: The critical variables we opt to track are defined as 3D center $[\tau_x, \tau_y, \tau_z]$, 3D dimensions $[\tau_w, \tau_h, \tau_l]$, orientation $[\tau_\theta, \tau_{\theta_h}]$, and scalar velocity τ_v. We define τ_θ as an θ orientation constrained to the range of $[-\frac{\pi}{2}, \frac{\pi}{2})$, and θ_h as in Sec. 3.1[3]. We constrain the motion model to only allow objects to move in the direction of their orientation. Hence, we define the state transition $\mathbf{F} \in \mathbb{R}^{9 \times 9}$ as:

---

[3] We do not use the axis θ_a of Sec. 3.1, since we expect the orientation to change *smoothly* and do not wish the orientation is *relative* to a (potentially) changing axis.

$$\mathbf{F} = \begin{bmatrix} & \cos(\tau_\theta + \pi\lfloor\tau_{\theta_h}\rceil) \\ & 0 \\ \mathbf{I}^{9\times 8} & -\sin(\tau_\theta + \pi\lfloor\tau_{\theta_h}\rceil) \\ & 0 \\ & \vdots \\ & 1 \end{bmatrix}, \tag{9}$$

where $\mathbf{I}$ denotes the identity matrix and the state variable order is respectively $[\tau_x, \tau_y, \tau_z, \tau_w, \tau_h, \tau_l, \tau_\theta, \tau_{\theta_h}, \tau_v]$. We constrain the velocity to only move within its orientation to simplify the Kalman to work more effectively. Recall that since our measurement model RPN processes a single frame, it does *not* measure velocity. Thus, to map the tracked state space and measurement space, we also define an observation model $\mathbf{H}$ as a truncated identity map of size $\mathbf{I} \in \mathbb{R}^{8\times 9}$. We define covariance $\mathbf{P}$ with 3D confidence μ, as $\mathbf{P} = \mathbf{I}^{9\times 9} \cdot (1-\mu) \cdot \lambda_o$ where λ_o is an uncertainty weighting factor. Hence, we avoid the need to manually tune the covariance, while being dynamic to diverse and changing image content.

Forecasting: The forecasting step aims to utilize the tracked state variables and covariances of time $t-1$ to estimate the state of a future time t. The equation to forecast a state variable τ_{t-1} into τ'_t is: $\tau'_t = \mathbf{F}_{t-1} \cdot \tau_{t-1}$, where $\mathbf{F}_{t-1}$ is the state transition model at $t-1$. Note that both objects *and* the capturing camera may have independent motion between consecutive frames. Therefore, we lastly apply the estimated ego-motion to all available tracks' 3D center $[\tau_x, \tau_y, \tau_z]$ by:

$$\begin{bmatrix} \tau_x \\ \tau_y \\ \tau_z \\ 1 \end{bmatrix}'_t = \begin{bmatrix} \mathbf{R}, & \mathbf{T} \\ 0, & 1 \end{bmatrix}^t_{t-1} \cdot \begin{bmatrix} \tau_x \\ \tau_y \\ \tau_z \\ 1 \end{bmatrix}'_t, \quad \tau'_{t\theta} = \tau'_{t\theta} + \rho_y \tag{10}$$

where $\mathbf{R}^t_{t-1} \in \mathbb{R}^{3\times 3}$ denotes the estimated rotation matrix converted from Euler angles and $\mathbf{T}^t_{t-1} \in \mathbb{R}^{3\times 1}$ the translation vector for ego-motion (as in Sec. 3.2). Finally, we forecast a tracked object's covariance $\mathbf{P}$ from $t-1$ to t defined as:

$$\mathbf{P}'_t = \mathbf{F}_{t-1} \cdot \mathbf{P}_{t-1} \cdot \mathbf{F}^T_{t-1} + \mathbf{I}^{9\times 9} \cdot (1-\mu_{t-1}), \tag{11}$$

where μ_{t-1} denotes the *average* self-balancing confidence μ of a track's life. Hence, the resultant track states τ'_t and track covariances $\mathbf{P}'_t$ represent the Kalman filter's best forecasted estimation with respect to frame t.

Association: After the tracks have been forecasted from $t-1$ to t, the next step is to associate tracks to corresponding 3D box measurements (Sec. 3.1). Let us denote boxes produced by the measurement RPN as $b \in \mathbb{R}^8$ mimicking the tracked state as $[b_x, b_y, b_z, b_w, b_h, b_l, b_\theta, b_{\theta_h}]^4$. Our matching strategy consists of two steps. We first compute the 3D center distance between the tracks τ'_t and measurements b. The best matches with the lowest distance are iteratively paired and removed until no pairs remain with distance $\leq k_d$. Then we compute the *projected* 2D box IoU between any remaining tracks τ'_t and measurements b.

---

[4] We apply the estimated transformations of Sec. 3.1 to their respective anchors with equations of [2], and backproject into 3D coordinates to match track variables.

The best matches with the highest IoU are also iteratively paired and removed until no pairs remain with IoU $\geq k_u$. Measured boxes that were *not* matched are added as new tracks. Conversely, tracks that were *not* matched incur a penalty with hyperparameter k_p, defined as $\mu_{t-1} = \mu_{t-1} \cdot k_p$. Lastly, any box who has confidence $\mu_{t-1} \leq k_m$ is removed from the valid tracks.

Update: After making associations between tracks τ_t' and measurements b, the next step is to utilize the track covariance $\mathbf{P}_t'$ and measured confidence μ to update each track to its final state τ_t and covariance $\mathbf{P}_t$. Firstly, we formally define the equation for computing the Kalman gain as:

$$\mathbf{K} = \mathbf{P}' \ \mathbf{H}^T \ (\mathbf{H} \ \mathbf{P}' \ \mathbf{H}^T + \mathbf{I}^{8\times8} \ (1-\mu) \cdot \lambda_o)^{-1}, \tag{12}$$

where $\mathbf{I}^{8\times8} \ (1-\mu) \cdot \lambda_o$ represents the incoming measurement covariance matrix, and $\mathbf{P}'$ the forecasted covariance of the track. Next, given the Kalman gain $\mathbf{K}$, forecasted state τ_t', forecasted covariance $\mathbf{P}_t'$, and measured box b, the final track state τ_t and covariance $\mathbf{P}_t$ are defined as:

$$\tau_t = \tau_t' + \mathbf{K} \ (b - \mathbf{H} \ \tau_t'), \qquad \mathbf{P}_t = (\mathbf{I}^{9\times9} - \mathbf{K} \ \mathbf{H}) \ \mathbf{P}_t'. \tag{13}$$

We lastly aggregate each track's overall confidence μ_t over time as a running average of $\mu_t = \frac{1}{2} \cdot (\mu_{t-1} + \mu)$, where μ is the measured confidence.

3.4   Implementation Details

Our framework is implemented in PyTorch [30], with the 3D RPN settings of [2]. We release source code at http://cvlab.cse.msu.edu/project-kinematic.html. We use a batch size of 2 and learning rate of 0.004. We set $k = 0.5$, $\lambda_o = 0.2$, $\lambda_r = 40$, $n_L = 100$, $\lambda_a = k_u = 0.35$, $k_d = 0.5$, $k_p = 0.75$, and $k_m = 0.05$. To ease training, we implement three phases. We first train the 2D-3D RPN with $L = L_{2D} + L_{3D}$, then the self-balancing loss of Eq. 7, for $80k$ and $50k$ iterations. We freeze the RPN to train ego-motion using L_{ego} for $80k$. Our backbone is DenseNet121 [17] where $n_h = 1,024$. Inference uses 4 frames as provided by [14].

4   Experiments

We benchmark our kinematic framework on the KITTI [14] dataset. We comprehensively evaluate on 3D Object Detection and Bird's Eye View (BEV) tasks. We then provide ablation experiments to better understand the effects and justification of our core methodology. We show qualitative examples in Fig. 6.

4.1   KITTI Dataset

The KITTI [14] dataset is a popular benchmark for self-driving tasks. The official dataset consists of 7,481 training and 7,518 testing images including annotations for 2D/3D objects, ego-motion, and 4 temporally adjacent frames. We evaluate on the most widely used validation split as proposed in [8], which consists of 3,712 training and 3,769 validation images. We focus primarily on the car class.

Metric: Average precision (AP) is utilized for object detection in KITTI. Following [38], the KITTI metric has updated to include 40 ($\uparrow 11$) recall points while *skipping* the first. The AP_{40} metric is more stable and fair overall [38]. Due to the official adoption of AP_{40}, it is not possible to compute AP_{11} on test. Hence, we elect to use the AP_{40} metric for all reported experiments.

	AP$_{3D}$ (IoU $\geq$ 0.7)			AP$_{BEV}$ (IoU $\geq$ 0.7)			s/im*
	Easy	Mod	Hard	Easy	Mod	Hard	
FQNet [25]	2.77	1.51	1.01	5.40	3.23	2.46	0.50†
ROI-10D [28]	4.32	2.02	1.46	9.78	4.91	3.74	0.20
GS3D [22]	4.47	2.90	2.47	8.41	6.08	4.94	2.00†
MonoPSR [19]	10.76	7.25	5.85	18.33	12.58	9.91	0.20
MonoDIS [38]	10.37	7.94	6.40	17.23	13.19	11.12	–
M3D-RPN [2]	14.76	9.71	7.42	21.02	13.67	10.23	*0.16*
AM3D [27]	*16.50*	*10.74*	**9.52**	*25.03*	*17.32*	**14.91**	0.40
Ours	**19.07**	**12.72**	*9.17*	**26.69**	**17.52**	*13.10*	**0.12**

Table 1. KITTI Test. We compare with SOTA methods on the KITTI test dataset. We report performances using the *updated* AP$_{40}$ metric from the official leaderboard. * the runtime is reported from the official leaderboard with slight variances in hardware. We indicate methods reported on CPU with $\dagger$. **Bold**/*italics* indicate best/second AP.

	AP$_{3D}$ (IoU $\geq$ [0.7/0.5])			AP$_{BEV}$ (IoU $\geq$ [0.7/0.5])		
	Easy	Mod	Hard	Easy	Mod	Hard
MonoDIS [38]	11.06/ –	7.60/ –	6.37/ –	18.45/ –	12.58/ –	10.66/ –
M3D-RPN [2]	*14.53/48.56*	*11.07/35.94*	*8.65/28.59*	*20.85/53.35*	*15.62/39.60*	*11.88/31.77*
Ours	**19.76/55.44**	**14.10/39.47**	**10.47/31.26**	**27.83/61.79**	**19.72/44.68**	**15.10/34.56**

Table 2. KITTI Validation. We compare with SOTA on KITTI validation [8] split. Note that methods published prior to [38] are unable to report the *updated* AP$_{40}$ metric.

4.2 3D Object Detection

We evaluate our proposed framework on the task of 3D object detection, which requires objects be localized in 3D camera coordinates as well as supplying the 3D dimensions and BEV orientation relative to the XZ plane. Due to the strict requirements of IoU in three dimensions, the task demands precise localization of an object to be considered a match (3D IoU $\geq$ 0.7). We evaluate our performance on the official test [14] dataset in Tab. 1 and the validation [8] split in Tab. 2.

We emphasize that our method improves the SOTA on KITTI test by a significant margin of $\uparrow$ 1.98% compared to [27] on the moderate configuration with IoU $\geq$ 0.7, which is the most common metric used to compare. Further, we note that [27] require multiple encoder-decoder networks which add overhead compared to our single network approach. Hence, their runtime is $\approx 3\times$ (Tab. 1) compared to ours, self-reported on *similar* but not identical GPU hardware. Moreover, [2] is the most comparable method to ours as both utilize a single network and an RPN archetype. We note that our method significantly outperforms [2] and many other recent works [19, 22, 25, 28, 38] by $\approx 3.01 - 11.21\%$.

We further evaluate our approach on the KITTI validation [8] split using the AP$_{40}$ for available approaches and observe similar overall trends as in Tab. 2. For instance, compared to competitive approaches [2, 38] our method improves the performance by $\uparrow$ 3.03% for the challenging IoU criteria of $\geq$ 0.7. Similarly, our performance on the more relaxed criteria of IoU $\geq$ 0.5 increases by $\uparrow$ 3.53%. We additionally visualize detailed performance characteristics on AP$_{3D}$ at discrete depth [15, 30, All] meters and IoU matching criterias $0.3 \rightarrow 0.7$ in Fig. 4.

	AP$_{3D}$ (IoU $\geq$ [0.7/0.5])			AP$_{BEV}$ (IoU $\geq$ [0.7/0.5])		
	Easy	Mod	Hard	Easy	Mod	Hard
Baseline	13.81/47.10	9.71/34.14	7.44/26.90	20.08/52.57	13.98/38.45	11.10/29.88
+ θ decomposition	16.66/51.47	12.10/38.58	9.40/30.98	23.15/56.48	17.43/42.53	13.48/34.37
+ self-confidence	16.64/52.18	12.77/38.99	9.60/**31.42**	24.22/58.52	18.02/42.95	13.92/**34.80**
+ $\mu = c \cdot \omega$	*18.28/54.70*	*13.55/39.33*	*10.13/31.25*	*25.72/60.87*	*18.82/44.36*	*14.48/34.48*
+ kinematics	**19.76/55.44**	**14.10/39.47**	**10.47**/*31.26*	**27.83/61.79**	**19.72/44.68**	**15.10**/*34.56*

Table 3. Ablation Experiments. We conduct a series of ablation experiments with the validation [8] split of KITTI, using diverse IoU matching criteria of $\geq 0.7/0.5$.

4.3 Bird's Eye View

The Bird's Eye View (BEV) task is similar to 3D object detection, differing primarily in that the 3D boxes are firstly projected into the XZ plane then 2D object detection is calculated. The projection collapses the Y-axis degree of freedom and intuitively results in a less precise but reasonable localization.

We note that our method achieves SOTA performance on the BEV task regarding the moderate setting of the KITTI test dataset as detailed in Tab. 1. Our method performs favorably compared with SOTA works [2, 19, 22, 25, 28, 38] (*e.g.*, $\approx 3.85 - 14.29\%$), and similarly to [27] at a notably lower runtime cost. We suspect that our method, especially the self-balancing confidence (Eq. 7), prioritizes precise localization which warrants more benefit in **full** 3D Object Detection task compared to the Bird's Eye View task.

Our method performs similarly on the validation [8] split of KITTI (Tab. 2). Specifically, compared to [2, 38] our proposed method outperforms by a range of $\approx 4.10 - 7.14\%$, which is **consistent** to the same methods on test $\approx 3.85 - 4.33\%$.

4.4 Ablation Study

To better understand the characteristics of our proposed kinematic framework, we perform a series of ablation experiments and analysis, summarized in Tab. 3. We adopt [2] without hill-climbing or depth-aware layers as our baseline method. Unless otherwise specified we use the experimental settings outlined in Sec. 3.4.

Orientation Improvement: The orientation of objects is intuitively a critical component when modeling motion. When the orientation is decomposed into axis, heading, and offset the overall performance significantly improves, *e.g.*, by $\uparrow 2.39\%$ in AP$_{3D}$ and $\uparrow 3.45\%$ in AP$_{BEV}$, as detailed within Tab. 3. We compute the mean angle error of our baseline, orientation decomposition, and kinematics method which respectively achieve $13.4°$, $10.9°$, and $6.1°$ ($\downarrow 54.48\%$), suggesting our proposed methodology is significantly more *stable*.

We compare our orientation decomposition to bin-based methods following general idea of [19, 22, 25, 28]. We specifically change our orientation definition into $[\theta_b, \theta_o]$ which includes a bin classification and an offset. We experiment with the number of bins set to [2, 4, 10] which are uniformly spread from $[0, 2\pi)$. Note that 4 bins have the same representational power as using binary $[\theta_a, \theta_h]$. We observe that the ablated bin-based methods achieve [9.47%, 10.02%, 10.76%] in AP$_{3D}$. In comparison, our decomposed orientation achieves 12.10% in AP$_{3D}$. We provide additional detailed experiments in our supplemental material.
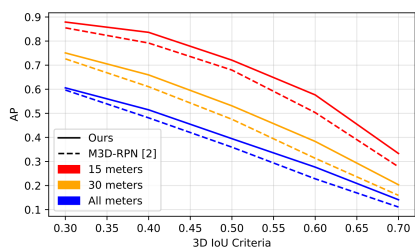
Fig. 4. We compare AP_{3D} with [2] by varying 3D IoU criteria *and* depth.
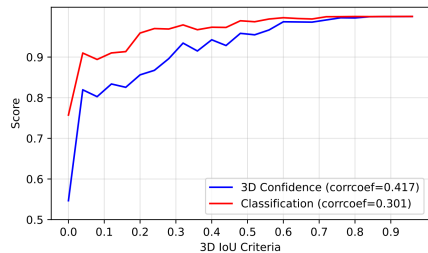
Fig. 5. We show the correlation of 3D IoU to classification c and 3D confidence μ.

Further, we find that our proposed kinematic motion model (as in Sec. 3.3) *degrades* in performance when a comparatively erratic baseline (Row 1. Tab. 3) orientation is utilized instead ($14.10 \rightarrow 11.47$ on AP_{3D}), reaffirming the importance of having a consistent/stable orientation when aggregating through time.

Self-balancing Confidence: We observe that the self-balancing confidence is important from two key respects. Firstly, its integration in Eq. 7 enables the network to re-weight box samples to focus more on reasonable samples and incur a flat penalty (*e.g.*, $\lambda_L \cdot (1 - \omega)$ of Eq. 7) on the difficult samples. In a sense, the self-balancing confidence loss is the *inverse* of hard-negative mining, allowing the network to focus on reasonable estimations. Hence, the loss on its own improves performance for AP_{3D} by $\uparrow 0.67\%$ and AP_{BEV} by $\uparrow 0.59\%$.

The second benefit of self-balancing confidence is that by design ω has an inherent correlation with the 3D object detection performance. Recall that we fuse ω with the classification score c to produce a final box rating of $\mu = c \cdot \omega$, which results in an additional gain of $\uparrow 0.78\%$ in AP_{3D} and $\uparrow 0.80\%$ in AP_{BEV}. We further analyze the correlation of μ with 3D IoU, as is summarized in Fig. 5. The correlation coefficient with the classification score c is significantly lower than the correlation using μ instead (0.301 vs. 0.417). In summary, the use of the Eq. 7 and μ account for a gain of $\uparrow 1.45\%$ in AP_{3D} and $\uparrow 1.39\%$ in AP_{BEV}.

Temporal Modeling: The use of video and kinematics is a significant motivating factor for this work. We find that the use of kinematics (detailed in Sec. 3.3) results in a gain of $\uparrow 0.55\%$ in AP_{3D} and $\uparrow 0.90\%$ in AP_{BEV}, as shown in Tab. 3. We emphasize that although the improvement is less dramatic versus orientation and self-confidence, the former are important to facilitate temporal modeling. We find that if orientation decomposition and uncertainty are **removed**, by using the baseline orientation and setting μ to be a static constant, then the kinematic performance drastically reduces from $14.10\% \rightarrow 10.64\%$ in AP_{3D}.

We emphasize that kinematic framework not only helps 3D object detection, but also *naturally* produces useful by-products such as velocity and ego-motion. Thus, we evaluate the respective average errors of each motion after applying the camera capture rate to convert the motion into miles per hour (MPH). We find that the per-object velocity and ego-motion speed errors perform reasonably at 7.036 MPH and 6.482 MPH respectively. We depict visual examples of all dynamic by-products in Fig. 6 and additionally in supplemental video.

Fig. 6. Qualitative Examples. We depict the image view (left) and BEV (right). We show velocity vector in green, speed and ego-motion in miles per hour (MPH) on top of detection boxes and at the top-left corner, and tracks as dots in BEV.

5   Conclusions

We present a novel kinematic 3D object detection framework which is able to efficiently leverage temporal cues and constraints to improve 3D object detection. Our method naturally provides useful by-products regarding scene dynamics, *e.g.*, reasonably accurate ego-motion and per-object velocity. We further propose novel designs of orientation estimation and a self-balancing 3D confidence loss in order to enable the proposed kinematic model to work effectively. We emphasize that our framework efficiently uses only a *single network* to comprehensively understand a highly dynamic 3D scene for urban autonomous driving. Moreover, we demonstrate our method's effectiveness through detailed experiments on the KITTI [14] dataset across the 3D object detection and BEV tasks.

References

1. Bertasius, G., Torresani, L., Shi, J.: Object detection in video with spatiotemporal sampling networks. In: ECCV. Springer (2018) 4
2. Brazil, G., Liu, X.: M3D-RPN: Monocular 3D region proposal network for object detection. In: ICCV. IEEE (2019) 1, 2, 3, 4, 5, 9, 10, 11, 12, 13
3. Brazil, G., Liu, X.: Pedestrian detection with autoregressive network phases. In: CVPR. IEEE (2019) 1
4. Brazil, G., Yin, X., Liu, X.: Illuminating pedestrians via simultaneous detection segmentation. In: ICCV. IEEE (2017) 1
5. Chabot, F., Chaouch, M., Rabarisoa, J., Teuliere, C., Chateau, T.: Deep MANTA: A coarse-to-fine many-task network for joint 2D and 3D vehicle analysis from monocular image. In: CVPR. IEEE (2017) 4
6. Chang, J.R., Chen, Y.S.: Pyramid stereo matching network. In: CVPR. IEEE (2018) 3
7. Chen, X., Kundu, K., Zhang, Z., Ma, H., Fidler, S., Urtasun, R.: Monocular 3D object detection for autonomous driving. In: CVPR. IEEE (2016) 3
8. Chen, X., Kundu, K., Zhu, Y., Berneshawi, A.G., Ma, H., Fidler, S., Urtasun, R.: 3D object proposals for accurate object class detection. In: NeurIPS (2015) 3, 10, 11, 12
9. Chen, Y., Liu, S., Shen, X., Jia, J.: Fast point R-CNN. In: ICCV. IEEE (2019) 1
10. Cordts, M., Omran, M., Ramos, S., Rehfeld, T., Enzweiler, M., Benenson, R., Franke, U., Roth, S., Schiele, B.: The cityscapes dataset for semantic urban scene understanding. In: CVPR. IEEE (2016) 4
11. Dollár, P., Wojek, C., Schiele, B., Perona, P.: Pedestrian detection: An evaluation of the state of the art. PAMI **34**(4), 743–761 (2012) 1
12. Felzenszwalb, P.F., Girshick, R.B., McAllester, D., Ramanan, D.: Object detection with discriminatively trained part-based models. PAMI **32**(9), 1627–1645 (2010) 3
13. Fu, H., Gong, M., Wang, C., Batmanghelich, K., Tao, D.: Deep ordinal regression network for monocular depth estimation. In: CVPR. IEEE (2018) 3
14. Geiger, A., Lenz, P., Urtasun, R.: Are we ready for autonomous driving? the KITTI vision benchmark suite. In: CVPR. IEEE (2012) 2, 3, 4, 10, 11, 14
15. Giancola, S., Zarzar, J., Ghanem, B.: Leveraging shape completion for 3D siamese tracking. In: CVPR. IEEE (2019) 4
16. Hu, H.N., Cai, Q.Z., Wang, D., Lin, J., Sun, M., Krahenbuhl, P., Darrell, T., Yu, F.: Joint monocular 3D vehicle detection and tracking. In: ICCV. IEEE (2019) 4
17. Huang, G., Liu, Z., Van Der Maaten, L., Weinberger, K.Q.: Densely connected convolutional networks. In: CVPR. IEEE (2017) 10
18. Ku, J., Mozifian, M., Lee, J., Harakeh, A., Waslander, S.L.: Joint 3D proposal generation and object detection from view aggregation. In: IROS. IEEE (2018) 3
19. Ku, J., Pon, A.D., Waslander, S.L.: Monocular 3D object detection leveraging accurate proposals and shape reconstruction. In: CVPR. IEEE (2019) 1, 3, 7, 11, 12
20. Kumar, A., Marks, T.K., Mou, W., Wang, Y., Jones, M., Cherian, A., Koike-Akino, T., Liu, X., Feng, C.: LUVLi face alignment: Estimating landmarks' location, uncertainty, and visibility likelihood. In: CVPR. IEEE (2020) 4
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. IEEE (2019) 1
22. Li, B., Ouyang, W., Sheng, L., Zeng, X., Wang, X.: GS3D: An efficient 3D object detection framework for autonomous driving. In: CVPR. IEEE (2019) 1, 3, 7, 11, 12

23. Liang, M., Yang, B., Chen, Y., Hu, R., Urtasun, R.: Multi-task multi-sensor fusion for 3D object detection. In: CVPR. IEEE (2019) 1
24. Liang, M., Yang, B., Wang, S., Urtasun, R.: Deep continuous fusion for multi-sensor 3D object detection. In: ECCV. Springer (2018) 1
25. Liu, L., Lu, J., Xu, C., Tian, Q., Zhou, J.: Deep fitting degree scoring network for monocular 3D object detection. In: CVPR. IEEE (2019) 1, 3, 4, 7, 11, 12
26. Liu, M., Zhu, M.: Mobile video object detection with temporally-aware feature maps. In: CVPR. IEEE (2018) 4
27. Ma, X., Wang, Z., Li, H., Zhang, P., Ouyang, W., Fan, X.: Accurate monocular 3D object detection via color-embedded 3d reconstruction for autonomous driving. In: ICCV. IEEE (2019) 1, 3, 4, 11, 12
28. Manhardt, F., Kehl, W., Gaidon, A.: ROI-10D: Monocular lifting of 2D detection to 6D pose and metric shape. In: CVPR. IEEE (2019) 1, 3, 7, 11, 12
29. Mousavian, A., Anguelov, D., Flynn, J., Kosecka, J.: 3D bounding box estimation using deep learning and geometry. In: CVPR. IEEE (2017) 3
30. Paszke, A., Gross, S., Chintala, S., Chanan, G., Yang, E., DeVito, Z., Lin, Z., Desmaison, A., Antiga, L., Lerer, A.: Automatic differentiation in pytorch (2017) 10
31. Qi, C.R., Liu, W., Wu, C., Su, H., Guibas, L.J.: Frustum pointnets for 3D object detection from RGB-D data. In: CVPR. IEEE (2018) 3
32. Qi, C.R., Yi, L., Su, H., Guibas, L.J.: PointNet++: Deep hierarchical feature learning on point sets in a metric space. In: NeurIPS (2017) 3
33. Redmon, J., Divvala, S., Girshick, R., Farhadi, A.: You only look once: Unified, real-time object detection. In: CVPR. IEEE (2016) 3, 4
34. 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. IEEE (2017) 4
35. Ren, S., He, K., Girshick, R., Sun, J.: Faster R-CNN: Towards real-time object detection with region proposal networks. In: NeurIPS (2015) 3, 4, 5
36. Russakovsky, O., Deng, J., Su, H., Krause, J., Satheesh, S., Ma, S., Huang, Z., Karpathy, A., Khosla, A., Bernstein, M., et al.: Imagenet large scale visual recognition challenge. IJCV **115**(3), 211–252 (2015) 4
37. Shi, S., Wang, X., Li, H.: PointRCNN: 3D object proposal generation and detection from point cloud. In: CVPR. IEEE (2019) 1
38. Simonelli, A., Bulo, S.R., Porzi, L., López-Antequera, M., Kontschieder, P.: Disentangling monocular 3D object detection. In: ICCV. IEEE (2019) 1, 3, 4, 10, 11, 12
39. Wang, X., Yin, W., Kong, T., Jiang, Y., Li, L., Shen, C.: Task-aware monocular depth estimation for 3D object detection. In: AAAI (2020) 3
40. Wang, Y., Chao, W.L., Garg, D., Hariharan, B., Campbell, M., Weinberger, K.Q.: Pseudo-LiDAR from visual depth estimation: Bridging the gap in 3D object detection for autonomous driving. In: CVPR. IEEE (2019) 1, 3
41. Welch, G., Bishop, G., et al.: An introduction to the kalman filter (1995) 2, 5, 8
42. Xiao, F., Jae Lee, Y.: Video object detection with an aligned spatial-temporal memory. In: ECCV. Springer (2018) 4
43. Xu, B., Chen, Z.: Multi-level fusion based 3D object detection from monocular images. In: CVPR. IEEE (2018) 3
44. 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. IEEE (2016) 4

45. Yang, Z., Sun, Y., Liu, S., Shen, X., Jia, J.: STD: Sparse-to-dense 3D object detector for point cloud. In: ICCV. IEEE (2019) 1, 4
46. Zhou, Y., Barnes, C., Lu, J., Yang, J., Li, H.: On the continuity of rotation representations in neural networks. In: CVPR. IEEE (2019) 4
47. Zhu, X., Dai, J., Yuan, L., Wei, Y.: Towards high performance video object detection. In: CVPR. IEEE (2018) 4
48. Zhu, X., Wang, Y., Dai, J., Yuan, L., Wei, Y.: Flow-guided feature aggregation for video object detection. In: ICCV. IEEE (2017) 4