Uncertainty-Aware Vehicle Orientation Estimation for Joint Detection-Prediction Models

Henggang Cui, Fang-Chieh Chou, Jake Charland, Carlos Vallespi-Gonzalez, Nemanja Djuric
Uber Advanced Technologies Group
{hcui2, fchou, jakec, cvallespi, ndjuric}@uber.com

Abstract

Object detection is a critical component of a self-driving system, tasked with inferring current states of the surrounding traffic actors. While there exist a number of studies on the problem of inferring the position and shape of vehicle actors, understanding actors' orientation remains a challenge for existing state-of-the-art detectors. Orientation is an important property for downstream modules of an autonomous system, particularly relevant for motion prediction of stationary or reversing actors where current methods struggle. We focus on this task and present a method that extends the existing joint detection-prediction models, allowing us to more accurately infer orientation of vehicles. In addition, the method is able to quantify prediction uncertainty, outputting the probability that the inferred orientation is flipped, which allows for improved motion prediction and safer autonomous operations. Empirical results show the benefits of the method, reaching state-of-the-art performance on the open-source nuScenes data set.

1 Introduction

In order to operate safely and efficiently in a real world, a self-driving vehicle (SDV) needs to be able to infer the current state of its surroundings, as well as to predict how this state would change in the near future. This task is addressed by object detection and motion prediction modules, two critical components of an autonomous system [1, 2, 3]. Recently researchers proposed to combine these two components into a unified, end-to-end model that achieved state-of-the-art performance [2, 4], which is the focus of our current work.

While achieving strong results, when it comes to vehicle detections most of the recent research has been focused on predicting objects' positions and bounding boxes. Nevertheless, object orientation (defined as a direction of the front of the vehicle) is an important information that the autonomous system requires to improve safety and efficiency, allowing better and more accurate future motion prediction. This is particularly true for static, slow-moving, and reversing actors, where common heuristic [5] that computes orientation from the travel direction breaks down.

Thus, to allow for accurate orientation prediction, common approach is to encode the orientation target $\phi \in (-180°, 180°]$ as a tuple $\{\sin(\phi), \cos(\phi)\}$ due to its circular nature. Then, orientation can be decoded using $\arctan\big(\sin(\phi), \cos(\phi)\big)$, which is referred to as *full-range representation*. An alternative is to use *half-range representation* [3] in applications where it is less important to distinguish the front and back of the bounding box. In these cases we can instead encode the orientation target as $\{\sin(2\phi), \cos(2\phi)\}$ and decode the orientation with $\arctan\big(\sin(2\phi), \cos(2\phi)\big)/2$, which gives the final output in the range $(-90°, 90°]$. Extending beyond the existing methods, we propose a novel representation and a loss that combines these ideas, resulting in improved performance.

We summarize the contributions of our work below:

- we study the trade-off between full-range representation and detection accuracy;
- we propose a method to represent full-range orientations without losing detection accuracy;
- the proposed method is able to estimate the probability that the orientation is flipped.

2 Related work

Estimating the full-range orientation is a challenging problem in LiDAR-based systems. This is mostly due to the fact that the front and back of a vehicle may look similar by considering just the LiDAR point cloud, especially when an actor is far away with few LiDAR points. To address this issue, several deep learning-based methods for object detection have been proposed that represent vehicle bounding boxes using half-range representation [4, 6]. Compared to models using the full-range representation [7], these methods lose the ability to distinguish the front and back of the vehicle, yet they are shown to achieve better detection accuracies thanks to the simplification of the task.

Beyond predicting the orientation itself, understanding its uncertainty is another important task that allows for safer autonomous operations. Authors of MultiBin [8] proposed an approach for handling the ambiguity of the orientation estimation. They proposed to bin the orientation into N overlapping bins and for each output two values, a confidence probability that the output angle lies within the bin and the residual rotation correction from the central angle. By doing that, the model can produce multiple orientation estimations for an actor, along with their probabilities. The method, however, requires one to tune the number of bins as well as their placement, whereas our method does not require extra hyper-parameters. Moreover, our work produces a full-range orientation along with its uncertainty without the discretization step, thus simplifying the learning problem. The evaluation results show that our proposed approach outperforms the state-of-the-art MultiBin method.

3 Methodology

While the proposed method is generic and can be applied to any model architecture, we implemented and evaluated it on top of MultiXNet [4], a state-of-the-art joint object detection and motion prediction model. The model takes as input the current and historical LiDAR point clouds along with a high-definition map of SDV's surroundings, which are rasterized onto a bird's-eye view (BEV) grid. The method then applies a multi-scale convolutional network on the resulting raster to detect objects' bounding boxes and predict their future trajectories for a total of H time steps. The bounding boxes are parameterized with center position, width and height dimensions, and orientation. For the orientation the model predicts the yaw component for all time steps, denoted as $\{\hat{\theta}_t\}_{t=1}^{H}$.

The original MultiXNet model uses the half-range orientation representation: $\{\sin(2\hat{\theta}_t), \cos(2\hat{\theta}_t)\}_{t=1}^{H}$ and the *half-range loss* given as

$$\mathcal{L}_{half} = \sum_{t=1}^{H} \ell_1\big(\sin(2\hat{\theta}_t) - \sin(2\theta_t)\big) + \ell_1\big(\cos(2\hat{\theta}_t) - \cos(2\theta_t)\big), \tag{1}$$

where ℓ_1 denotes the smooth-L1 loss and $\{\sin(2\theta_t), \cos(2\theta_t)\}_{t=1}^{H}$ come from the ground-truth labels.

3.1 Combining half-range and full-range losses

To represent the orientations in the full $360°$ range, the orientation target can be encoded in a straightforward way as $\{\sin(\hat{\theta}_t), \cos(\hat{\theta}_t)\}_{t=1}^{H}$ and trained with the *full-range loss*, given as

$$\mathcal{L}_{full} = \sum_{t=1}^{H} \ell_1\big(\sin(\hat{\theta}_t) - \sin(\theta_t)\big) + \ell_1\big(\cos(\hat{\theta}_t) - \cos(\theta_t)\big). \tag{2}$$

However, our evaluation results show that the full-range representation leads to lower model performance, both in detection and prediction metrics. To address this problem, we propose to add to (2) an additional half-range loss term from (1) to train the model, where the half-range representation can be computed from the full-range representation using the following trigonometric identities,

$$\sin(2\hat{\theta}_t) = 2\sin(\hat{\theta}_t)\cos(\hat{\theta}_t), \quad \cos(2\hat{\theta}_t) = \cos^2(\hat{\theta}_t) - \sin^2(\hat{\theta}_t). \tag{3}$$

Evaluation results show that combining the losses significantly improves model performance.

3.2 Flipping-aware orientation prediction

Even with the combined loss, the model may still incur a high loss from the full-range loss component when predicting a flipped bounding box. To further mitigate this problem, we propose a novel *flipping-aware loss*. In particular, in addition to orientation outputs $\{\sin(\hat{\theta}_t), \cos(\hat{\theta}_t)\}_{t=1}^{H}$ the model is also trained to predict a probability that its orientation predictions are flipped by $180°$, denoted by $\hat{p}_f$. To achieve this we define a *flipped full-range loss* as follows,

$$\mathcal{L}_{flipped} = \sum_{t=1}^{H} \ell_1\big(-\sin(\hat{\theta}_t) - \sin(\theta_t)\big) + \ell_1\big(-\cos(\hat{\theta}_t) - \cos(\theta_t)\big). \tag{4}$$

Then, for each actor we compare the values of $\mathcal{L}_{full}$ and $\mathcal{L}_{flipped}$ and use $p_f = \mathbb{1}_{\mathcal{L}_{full} > \mathcal{L}_{flipped}}$ as the ground-truth classification label to train the flipped classification output $\hat{p}_f$, where $\mathbb{1}_{cond}$ is an indicator function equaling 1 if a condition $cond$ is true and 0 otherwise.

Lastly, we define the final loss as

$$\mathcal{L}_{final} = \mathcal{L}_{half} + \min(\mathcal{L}_{full}, \mathcal{L}_{flipped}) + \text{CrossEntropy}(\hat{p}_f, \mathbb{1}_{\mathcal{L}_{full} > \mathcal{L}_{flipped}}). \tag{5}$$

Note that the loss penalizes only the minimum of $\mathcal{L}_{full}$ and $\mathcal{L}_{flipped}$. Thus, if the model predicts a flipped orientation for the bounding box, it will not be penalized by the orientation loss but by the flipped classification loss instead, unless it predicts a high flipped probability. As a result, when the model predicts a flipped orientation it will be encouraged to keep pushing the orientation closer to the flipped ground-truth orientation, while moving the flipped probability closer to 1. Since the model only needs to predict one extra orientation flipped probability value for each actor, our method has a negligible impact on the model inference speed. Lastly, following the training completion we implement a post-processing step to flip the orientations whose probabilities are greater than 0.5, and update their flipped probabilities as $\hat{p}_f \leftarrow 1 - \hat{p}_f$.

4 Evaluation

4.1 Experimental setup

We evaluated our method on the nuScenes [9] data set, which contains $1{,}000$ scenes with $390{,}000$ LiDAR frames. Our MultiXNet implementation uses the following hyper-parameters (see [4] for the exact definitions): $L = 100$m, $W = 100$m, $V = 8$m, $\Delta_L = 0.125$m, $\Delta_W = 0.125$m, $\Delta_V = 0.2$m for the BEV input size and resolution, $T = 10$ at 20Hz for the historical LiDAR inputs, $H = 30$ at 10Hz for future state predictions (in other words, using 1s of history to predict 3s into future).

We used the standard KITTI [10] object detection and prediction metrics, including Average Orientation Similarity (AOS), Average Precision (AP), orientation errors, and trajectory ℓ_2 error, evaluated on vehicle actors. Following [4] we use an IoU threshold of 0.7 when computing the AP; note that as AP is IoU-based a flipped bounding box will still be counted as a true positive. The AOS metric weights the precision by the average cosine distance (normalized to a $[0, 1]$ range) of the orientations at each recall point. As a result, the AP metric is by definition an upper bound of AOS, and a completely flipped bounding box will have no positive contribution towards AOS. We also measure the orientation error of the bounding boxes at 0s and the future trajectory ℓ_2 error at 3s of the true positive detections, with the operating point set at 0.8 recall. For the orientation error we measure both full-range (FOE) and half-range orientation error (HOE), and also slice FOE by moving and non-moving actors. The FOE and HOE metrics are defined as follows,

$$\text{FOE} = |(\theta_0 - \hat{\theta}_0) \mod 360°|, \quad \text{HOE} = |(\theta_0 - \hat{\theta}_0) \mod 180°|. \tag{6}$$

We compared our `Flip-aware` method against the `Half-repr` method used by [3, 4, 6], the `Full-repr` method used by [7], and the MultiBin method [8]. We denote MultiBin with n bins as `MultiBin-n`, with the bins centered at $\{0°, 180°\}$ for $n = 2$ and $\{-90°, 0°, 90°, 180°\}$ for $n = 4$. To fairly evaluate FOE for `Half-repr`, we used the direction of the predicted trajectory to convert the corresponding orientation to the full range.

Table 1: Comparison of the competing methods

			Orientation error [deg] $\downarrow$			
			Half-range	**Full-range**		
Method	**AOS$_{0.7}$** $\uparrow$	**AP$_{0.7}$** $\uparrow$	**All**	**All**	**Moving**	ℓ_2**@3s [m]** $\downarrow$
`Half-repr`	40.7 ± 0.7	$\mathbf{60.8 \pm 1.0}$	$\mathbf{1.72 \pm 0.04}$	59.9 ± 1.4	4.7 ± 0.5	$\mathbf{0.99 \pm 0.02}$
`Full-repr`	55.1 ± 0.5	57.1 ± 0.5	2.32 ± 0.01	$\mathbf{8.2 \pm 0.2}$	2.4 ± 0.2	1.01 ± 0.02
`MultiBin-2`	55.0 ± 0.3	57.3 ± 0.6	2.54 ± 0.01	9.4 ± 0.7	3.0 ± 0.1	1.01 ± 0.01
`MultiBin-4`	55.5 ± 0.3	58.0 ± 0.5	2.14 ± 0.07	9.8 ± 0.5	2.6 ± 0.3	1.01 ± 0.01
`Flip-aware`	$\mathbf{57.9 \pm 0.4}$	$\mathbf{60.7 \pm 0.2}$	$\mathbf{1.71 \pm 0.04}$	9.6 ± 0.8	$\mathbf{2.2 \pm 0.1}$	$\mathbf{0.99 \pm 0.01}$

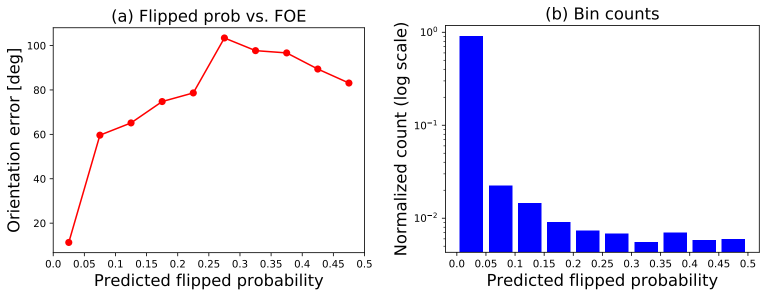

Figure 1: Analysis of the orientation uncertainty outputs of the proposed `Flip-aware` method

4.2 Quantitative results

Comparison of different methods on the nuScenes data is given in Table 1. We trained all models three times and report the means and standard deviations of the metrics, marking the best-performing methods in bold. We can see that `Half-repr` had the best AP and ℓ_2 performance, however it was not able to represent full-range orientations as seen by large orientation errors. Even when using the trajectory prediction to infer the orientations FOE was still very high, especially when it comes to the static actors, and it also exhibited very low AOS. `Full-repr` had the lowest FOE, but it had the worst AP and ℓ_2 performance, trailing `Half-repr` by a large margin (showing almost 5% drop in AP). This could be explained by the fact that the network is incurring high losses for predicting a flipped bounding box, which causes the model to not spend enough learning capacity in accurately estimating the half-range orientations. The `MultiBin-n` models had similar performance compared to `Full-repr`. We hypothesize that MultiBin is not able to achieve better performance for our application as it was designed for 3D orientation estimation from camera images, while we estimate the orientations from BEV voxels. Our proposed `Flip-aware` method achieved significant improvement in AP and HOE compared to the other full-range representations, and it achieved the best AOS, AP, HOE, moving FOE, and ℓ_2 metrics among all the methods.

Figure 1 presents the analysis of uncertainty outputs of the proposed `Flip-aware` method. We binned all actors by their predicted flipped probabilities, and for each bin we report the average FOE and actor counts. Results in Figure 1(a) show that when the model predicts an actor to have a low flipped probability, its FOE is also expected to be significantly lower than for those actors that have higher probabilities. We can conclude that the predicted flipped probability is strongly correlated with the expected error, and is a reliable measure of our uncertainty in the actor's orientation. From Figure 1(b), we can see that the model outputs low uncertainty for most of the actors, mirroring the distribution of the actor speeds in the data set.

5 Conclusion

In this paper we considered the problem of object detection and motion prediction in the context of self-driving technology. Unlike earlier work, we focused on the subtask of orientation prediction for vehicle actors. This is critical for a full understanding of the actors' state and their motion prediction, especially for slow-moving and stopped vehicles. In addition to improved orientation prediction, the proposed approach also quantifies the prediction uncertainty by inferring the flipped probability, which is very useful for downstream modules in a self-driving system. Experiments on the real-world, open-source nuScenes data set indicate the benefits of the proposed method.

References

[1] C. Urmson *et al.*, "Self-driving cars and the urban challenge," *IEEE Intelligent Systems*, vol. 23, no. 2, pp. 66–68, 2008.

[2] S. Casas, W. Luo, and R. Urtasun, "Intentnet: Learning to predict intention from raw sensor data," in *Conference on Robot Learning*, 2018, pp. 947–956.

[3] G. P. Meyer, A. Laddha, E. Kee, C. Vallespi-Gonzalez, and C. K. Wellington, "LaserNet: An efficient probabilistic 3D object detector for autonomous driving," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2019.

[4] N. Djuric, H. Cui, Z. Su, S. Wu, H. Wang, F.-C. Chou, L. S. Martin, S. Feng, R. Hu, Y. Xu *et al.*, "Multixnet: Multiclass multistage multimodal motion prediction," *arXiv preprint arXiv:2006.02000*, 2020.

[5] H. Cui, T. Nguyen, F.-C. Chou, T.-H. Lin, J. Schneider, D. Bradley, and N. Djuric, "Deep kinematic models for kinematically feasible vehicle trajectory predictions," in *2020 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2020, pp. 10 563–10 569.

[6] B. Yang, M. Liang, and R. Urtasun, "Hdnet: Exploiting hd maps for 3d object detection," in *Conference on Robot Learning*, 2018, pp. 146–155.

[7] W. Luo, B. Yang, and R. Urtasun, "Fast and furious: Real time end-to-end 3d detection, tracking and motion forecasting with a single convolutional net," in *Proceedings of the IEEE conference on Computer Vision and Pattern Recognition*, 2018, pp. 3569–3577.

[8] A. Mousavian, D. Anguelov, J. Flynn, and J. Kosecka, "3d bounding box estimation using deep learning and geometry," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2017, pp. 7074–7082.

[9] H. Caesar, V. Bankiti, A. H. Lang, S. Vora, V. E. Liong, Q. Xu, A. Krishnan, Y. Pan, G. Baldan, and O. Beijbom, "nuscenes: A multimodal dataset for autonomous driving," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2020, pp. 11 621–11 631.

[10] A. Geiger, P. Lenz, C. Stiller, and R. Urtasun, "Vision meets robotics: The kitti dataset," *The International Journal of Robotics Research*, vol. 32, no. 11, pp. 1231–1237, 2013.