Distance-Aware Single-Stage Detectors: Combining Detection with Object-Specific Distance Estimation

Jon Hernandez Aranda
School of Computing
KAIST
Daejeon, Republic of Korea
jonher16@kaist.ac.kr

Patrick Dominique Vibild
School of Computing
KAIST
Daejeon, Republic of Korea
patrickvibild@gmail.com

Daeyoung Kim
School of Computing
KAIST
Daejeon, Republic of Korea
kimd@kaist.ac.kr

***Abstract*—Monocular distance estimation enables 3D understanding from single-camera images, supporting navigation and safety applications. We present two RetinaNet-based single-stage models: DistinaNet and DistinaNet-BBoxExt, enhancing the RetinaNet architecture with distance estimation capabilities via ExtraHead and BBoxExtension strategies. Benchmarked on the KITTI dataset, DistinaNet outperforms state-of-the-art object-specific methods by 44.82% in MAE. Ablation studies show that adding the distance estimation task does not degrade detection performance. We also analyze FPN layer contributions, head architectures, and loss functions. Both models demonstrate strong generalization to unseen real-world driving scenarios, validated using data captured from a custom data collection pipeline. Code is available at https://github.com/jonher16/distinanet for reproduction. *Index Terms*—Distance Estimation, Depth Estimation, Object Detection, Single-Stage Architecture, Multi-Task Learning**

I. Introduction

Distance estimation determines spatial separation between objects in a scene. This is a foundation for interpreting and reconstructing three-dimensional information from two-dimensional inputs.

Active methods, like Radar and LiDAR, measure distances using ultrasound or laser emissions [1].

Passive methods are more cost-effective and estimate distances using camera images and complex algorithms [2]. We can distinguish between two kinds of *passive* methods:

Stereo Vision employs two cameras for depth estimation but suffers increasing errors with distance due to the inverse proportionality of disparity to depth [3].

Monocular Vision uses a single camera, reducing costs and enabling integration in systems with size, weight, and power constraints. It plays a vital role in transportation, enhancing advanced driver-assistance systems with real-time data [4], and autonomous UAV navigation, where researchers have achieved 95% accuracy in detecting airborne objects within 700 meters [5].

Modern approaches leverage deep learning, particularly Convolutional Neural Networks (CNNs), using annotated datasets of objects and their distances for improved accuracy [2], [4]–[15].

Deep-learning based monocular distance estimation techniques can be grouped into three categories [16]:

2D Detection + Depth Estimation generates depth maps from RGB images using encoder-decoder architectures, integrated with detection [15] or segmentation models [17]. However, it is limited to around 100 meters in outdoor environments [18].

3D Object Detection creates three-dimensional bboxes (bounding boxes) around objects but requires additional 3D bbox annotations, increasing labeling costs [19], [20].

Object-Specific Distance Estimation uses a multi-task strategy, simultaneously predicting object bboxes and distances, reducing data acquisition costs and complexity [21], [22].

Existing research often introduces models designed for specific applications of monocular distance estimation without comparing alternative deep learning techniques and models used for distance estimation. Many proposed models depend solely on custom private datasets [2], [10], which obstructs reproducibility and frequently does not provide accessible code or enough details for reproduction [2], [4]–[9], [11], [12], further complicating the replication of results.

This paper presents two single-stage detector-based distance estimation models trained on the KITTI dataset [23]. Our models outperform all current SO single-stage models benchmarked on the KITTI dataset [6], [7], [9], [11], [16], [21]. We propose single-stage detectors due to their faster inference times compared to multi-stage models, making them suitable for deployment in real-time applications. The proposed models are benchmarked against other object-specific state-of-the-art models, and we provide qualitative results on unseen data. Additionally, we analyze different techniques for integrating distance estimation into object detection, conduct ablation studies on key components, and validate generalization with real-world data.

We summarize our main contributions as follows:

1) We introduce the first RetinaNet-based single-stage models for object-specific distance estimation: **DistinaNet** and **DistinaNet-BBoxExt**, achieving significant performance gains in KITTI. Code is publicly released.

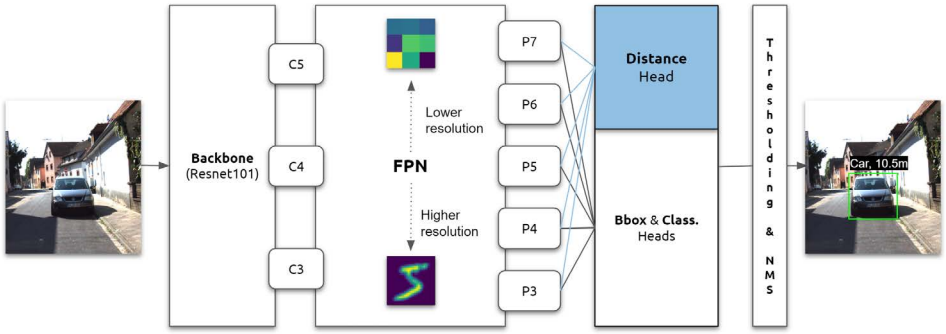

Fig. 1. Architecture of DistinaNet.

2) We conduct extensive ablation studies to identify and integrate the most impactful architectural components for distance estimation.
3) We validate model generalization on real-world urban scene dataset created using a custom data collection pipeline.

II. Related Works

Object-specific monocular distance estimation models can be broadly categorized into two architectures: *single-stage* and *multi-stage*, as illustrated in Fig. 2. Single-stage models predict object locations and distances in one pass, offering speed and simplicity, whereas multi-stage models use separate modules for detection and regression, often achieving higher accuracy at the cost of greater complexity.

Most existing methods adopt multi-stage approaches, where pre-trained detectors like YOLO are combined with MLPs or CNNs for distance regression [2], [9]. Some methods crop detected objects before feature extraction and regression [10], [11], while others stack multiple modules in compound pipelines involving detection, alignment, and validation stages (*Compound Architectures*) [5], [6], [16].

In contrast, few works adopt single-stage architectures. Dist-YOLO [7] extends the bbox head to regress distance (*BBox Head Extension technique*), reusing detection features.

Despite these advances, single-stage models remain under-explored. In this work, we introduce two novel RetinaNet-based single-stage detectors and benchmark their performance against prior state-of-the-art methods.

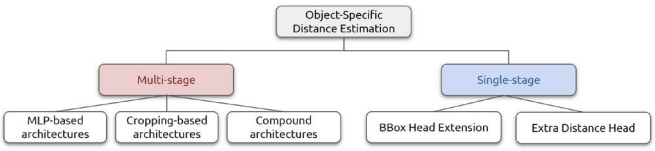

Fig. 2. Taxonomy of the presented related works in object-specific monocular distance estimation.

III. Methodology

A. DistinaNet

DistinaNet builds on RetinaNet [24], a well-established detector known for strong performance across diverse tasks [25]–[28]. Its modular design makes it suitable for analyzing architectural components relevant to distance estimation. While we use RetinaNet, the proposed methodology is applicable to any single- or multi-stage detector.

DistinaNet integrates an extra distance estimation head into the RetinaNet architecture, as shown in Fig. 1. The input image is processed through a ResNet101 backbone, extracting the last three feature maps at different scales. These maps are refined by the Feature Pyramid Network (FPN), producing five outputs: P_3 to P_7. Each FPN layer generates priors matched to objects in the image, which are fed into three parallel heads for bbox regression, classification, and distance estimation. Final predictions are obtained via thresholding and Non-Maximum Suppression (NMS).

The bbox regression head output per feature map has the following shape:

$$[B, H_i \times W_i \times A, 4] \tag{1}$$

where B is the batch size, H_i and W_i are the height and width of the feature map, A is the number of anchor boxes per feature map location, and the 4-channel output per anchor consists of $[t_x, t_y, t_w, t_h]$, representing bbox transformation parameters.

The output shape from the distance estimation head is:

$$[B, H_i \times W_i \times A, 1] \tag{2}$$

1) Distance Estimation Head Architectures: We design five distance head architectures, illustrated in Fig. 3. The baseline *BaseConv* mirrors the bbox regression head, using four convolutional layers followed by ReLU, with the final layer matching the anchor dimensions. *DeepConv* extends this with two additional layers to explore deeper feature extraction. *Bottleneck* compresses and expands feature dimensions via a bottleneck block, reducing computation and potential overfitting. *CBAM* [29] introduces attention by adding four CBAM layers after the convolutional stack, enhancing feature representation. Finally, *Dynamic Branching* leverages a

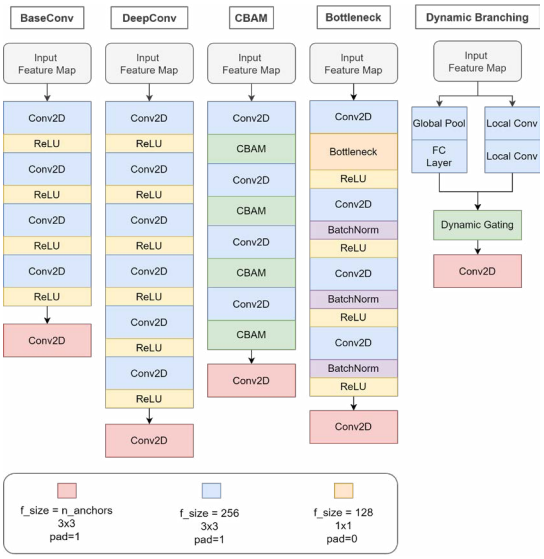

Fig. 3. Distance Head Architectures of DistinaNet.

dynamic gating block [30] to adaptively combine local and global features based on the input.

2) Distance Estimation Loss Functions: To integrate distance estimation alongside the Focal Loss [24], we evaluate several loss functions for regression, analyzing their effect on model learning and performance.

The *L1 loss* measures absolute error and is our default baseline. *L2 loss* penalizes large errors more heavily, making it more sensitive to outliers. *Huber Loss* [31], shown in Eq. 3, combines the robustness of L1 with the precision of L2, using $\delta = 0.5$. *Smooth L1*, defined in Eq. 4, is a variation of Huber with $\delta = 1$. *Log-Cosh* loss [31] (Eq. 5) approximates L2 for small errors and L1 for large ones, offering stable gradients.

$$Huber(y, \hat{y}, \delta) = \begin{cases} 0.5(y_i - \hat{y}_i)^2 & \text{if } |y_i - \hat{y}_i| \leq \delta \\ \delta(|y_i - \hat{y}_i| - 0.5\delta) & \text{otherwise} \end{cases} \tag{3}$$

$$SmoothL1(y, \hat{y}) = \begin{cases} 0.5\frac{(y_i - \hat{y}_i)^2}{\delta} & \text{if } |y_i - \hat{y}_i| < \delta \\ |y_i - \hat{y}_i| - 0.5\delta & \text{otherwise} \end{cases} \tag{4}$$

$$LogCosh(y, \hat{y}) = \sum_{i=1}^{n} \log(\cosh(y_i - \hat{y}_i)) \tag{5}$$

The final training loss combines classification, bbox regression, and distance losses:

$$L_t = L_{cls} + L_{reg} + \gamma \cdot L_d \tag{6}$$

where γ balances the importance of the distance loss. We experiment with different γ values to optimize joint performance across tasks. .

B. DistinaNet-BBoxExt

In addition to DistinaNet, we used the RetinaNet base detector to implement the *BBox Head Extension technique* and compare its effectiveness. In this approach, instead of adding an extra head capable of learning specific features for regressing distance, we extend the bbox prediction vector to include the distance value as a prediction that shares the weights between both tasks.

Therefore, using this technique, the output shape of the bbox regression head becomes:

$$[B, H_i \times W_i \times A, 5] \tag{7}$$

The 5-channel output per anchor consists of $[t_x, t_y, t_w, t_h, d]$, representing bbox transformation parameters with the extension of the distance value, d. The output vector of the classification head remains the same.

C. Object-Specific Data Collection Pipeline

To evaluate generalization beyond KITTI, we developed an object-specific data collection pipeline that captures and annotates urban road scenes with different visual distributions. We use a ZED X camera (4mm focal length, 1920×1080 resolution) mounted on a vehicle and a 16GB RAM NVIDIA Jetson Orin NX for on-device processing.

As shown in Fig. 4, images are passed through YOLOv9 [32] to generate bboxes, which are annotated with depth values at each box center. We collected 100 such images to qualitatively evaluate DistinaNet and DistinaNet-BBoxExt on previously unseen data. Unlike KITTI's test set, our custom data helps assess model robustness under domain shift and reduces overfitting risks.

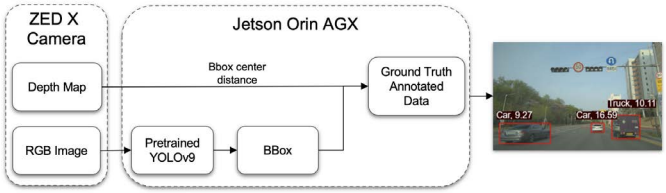

Fig. 4. Illustration of our object-specific monocular distance estimation dataset generation pipeline.

IV. Experiments

We conducted experiments on a system with 8 NVIDIA A100-SXM4 GPUs and dual AMD EPYC 7742 processors. Models were implemented using Python 3.10.13, PyTorch 1.13.1, and CUDA 11.7, with a fixed seed value of 16 for reproducibility.

1) Dataset: The KITTI dataset [23] is a widely used benchmark for object-specific distance estimation. It includes 7481 labeled and 7518 unlabeled images across nine object classes, with 'Car' being the most frequent. Object distances range from 0 to 150 meters, although most fall within 5–80 meters.

TABLE I
COMPARISON OF DISTANCE PERFORMANCE METRICS FOR DIFFERENT OBJECT-SPECIFIC MONOCULAR DISTANCE ESTIMATION MODELS

Model	MAE (↓)	MRE (↓)	RMSE (↓)	RMSLE (↓)
CNN-ROI (Zhu *et al.* 2019) [6]	-	0.251	6.870	0.314
Dist-YOLO (Vajgl *et al.* 2022)†	3.851	0.237	4.876	0.293
CL-CSEN (Ahishali *et al.* 2021) [11]	-	0.193	4.085	0.2604
YOLOv7-CBAM (Afshar *et al.* 2023) [9]	1.700	0.170	-	-
Dist-YOLO (Vajgl *et al.* 2022) [7]	2.570	0.110	-	-
Crop-VGG16 (Lai *et al.* 2020)* [10]	1.786	0.096	2.815	0.129
YOLOv8-MLP*	1.554	0.078	3.077	<u>0.114</u>
YOLOv8n-DeepSort (Song *et al.* 2024) [21]	1.249	<u>0.043</u>	1.924	-
RRN (Zhang *et al.* 2020) [16]	<u>1.169</u>	0.053	<u>1.897</u>	-
DistinaNet-BBoxExt (ours)	**0.663**	**0.030**	**1.248**	**0.053**
DistinaNet (ours)	**0.645**	**0.030**	**1.200**	**0.053**

Tested on the KITTI test set. The arrows indicate the preferred trend for each metric. Bold denotes the best results, and <u>underline</u> denotes the second-best results. * signifies results from our implementation of the model. † denotes results from the authors' implementation that we reproduced. Some metrics were not provided in the referenced papers.

We split the labeled set into 70% training, 10% validation, and 20% test subsets.

Most prior works [7], [9], [11], [15], [16], [21], [33] rely solely on KITTI for both training and evaluation. Although some studies [7], [10] test on private datasets to assess generalization, differences in camera intrinsics often limit fair comparisons and degrade model performance under domain change.

2) Evaluation Metrics: We evaluate distance estimation using standard regression metrics: Mean Absolute Error (MAE), Root Mean Square Error (RMSE), Mean Relative Error (MRE), and Root Mean Square Logarithmic Error (RMSLE). MAE and RMSE measure absolute differences, while MRE accounts for scale, and RMSLE balances large and small prediction errors.

For object detection, we use Mean Average Precision (mAP), computed as the mean of class-wise Average Precision (AP), which summarizes the precision-recall curve using recall-weighted precision scores.

A. Benchmark

We benchmark our proposed DistinaNet and DistinaNet-BBoxExt models against state-of-the-art object-specific distance estimation methods, including Dist-YOLO [7] (MLP-based regression from YOLOv9 outputs) and Crop-VGG16 [10].

Both models are trained for 200 epochs with early stopping (patience = 20), using a ResNet101 backbone, batch size of 1, learning rate 5×10^{-5}, Adam optimizer, and Huber Loss ($\delta = 0.5$). DistinaNet uses the Bottleneck distance head.

As shown in Table I, DistinaNet achieves the best performance across all distance metrics, with DistinaNet-BBoxExt close behind. Compared to the RNN model, DistinaNet improves absolute error by 44.82%, and DistinaNet-BBoxExt by 43.2%, confirming the effectiveness of single-stage approaches.

DistinaNet runs at 0.037 seconds per inference, though inference time is omitted from the table due to lack of reporting in prior works.

B. Ablation Studies on DistinaNet

We perform ablation studies on DistinaNet to assess the contribution of key components. Specifically, we evaluate: (1) the effect of multi-task learning on detection and distance performance, (2) the role of different FPN layers across distance ranges, (3) various distance head architectures, and (4) alternative loss functions.

All ablations use consistent training settings: 200 epochs, ResNet101 backbone, batch size of 1, learning rate of 1×10^{-5}, and the Adam optimizer. Unless stated otherwise, the distance head defaults to the *BaseConv* architecture (shared with the bbox head), and the loss function is L1.

1) Multi-Task Learning: We evaluate the effect of multi-task learning (MTL) by comparing the original RetinaNet to DistinaNet, which adds distance estimation as a secondary task. Additionally, we test DistinaNet variants with different distance loss weights (γ) to assess the trade-off between detection and distance estimation performance.

As shown in Table II, incorporating distance estimation consistently maintains or improves detection accuracy. With $\gamma = 10$, mAP improves by 1.75% over the baseline. For distance metrics, $\gamma = 1$ yields the best MAE and MRE, while $\gamma = 0.1$ slightly improves RMSE and RMSLE. Very high weights (e.g., 100) degrade overall performance, as the model overfocuses on distance at the expense of detection. Moderate weights (0.1–1) offer the best balance.

TABLE II
EVALUATION METRICS FOR THE DISTINANET MODEL WITH DIFFERENT DISTANCE LOSS WEIGHTS

Distance Loss Weight (γ)	mAP (↑)	MAE (↓)	MRE (↓)	RMSE (↓)	RMSLE (↓)
100	0.774	0.853	0.037	1.821	0.071
10	**0.811**	0.771	0.034	1.593	0.065
1	0.806	**0.678**	**0.031**	1.305	0.056
0.1	0.797	0.682	**0.031**	**1.293**	**0.050**
0.01	0.797	0.792	0.037	1.426	0.057
0	0.797	-	-	-	-

Tested in the KITTI test set. The arrows indicate the preferred trend for each metric. Bold values mean the best performance.

2) FPN: We analyze the contribution of each FPN layer (P3–P7) in DistinaNet by training variants with different layer combinations: 3-4-5 (high resolution), 5-6-7 (low resolution), and 3-7 (full range). This helps assess accuracy vs. model complexity trade-offs.

Fig. 5 shows MAE across distance ranges for each layer when all FPN layers are active. Higher-resolution layers (e.g., P3–P4) are more effective at short ranges, while lower-resolution layers handle broader distances. Layer 3 is the most consistent, even predicting beyond 60 meters. It does not contribute to 0–5m predictions due to anchor size. Layer 7 provides no detections, suggesting it can be removed to reduce computation without loss in performance.

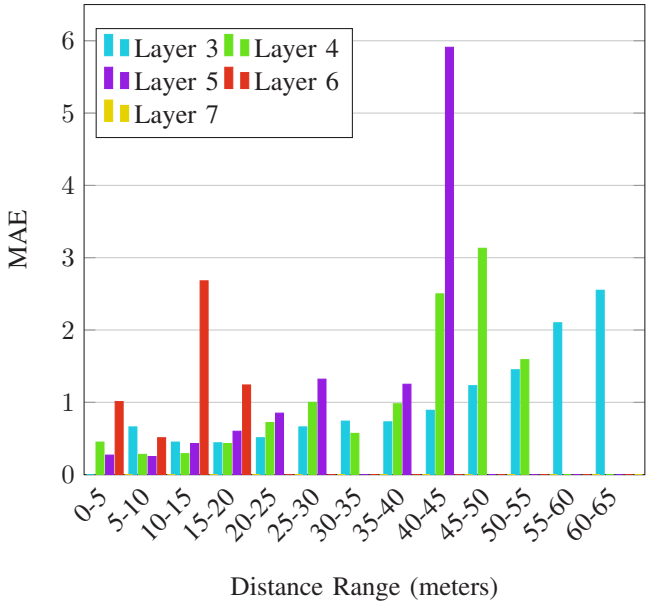

Fig. 5. MAE of FPN Layers at different distance ranges.

Table III summarizes performance across different FPN layer groups. The 3-4-5 configuration achieves the best overall results, showing that a reduced set of higher-resolution layers can improve accuracy while simplifying the model. Although Layer 6 contributes some detections, it introduces notable distance errors, and Layer 7 has no effect. The 5-6-7 group yields low MAE but poor detection, likely due to the dominance of nearby objects on KITTI. The full 3-7 setup performs moderately, with Layer 3 as the key contributor.

TABLE III
EVALUATION METRICS FOR THE DISTINANET MODEL WITH DIFFERENT FPN LAYER CONFIGURATIONS

FPN Layer Group	mAP (↑)	MAE (↓)	MRE (↓)	RMSE (↓)	RMSLE (↓)
FPN 3-4-5-6-7	**0.806**	0.678	0.031	1.305	0.056
FPN 3-4-5	0.800	**0.663**	**0.030**	**1.297**	**0.050**
FPN 5-6-7	0.241	0.416	0.045	0.766	0.069
FPN 3-7	0.497	0.857	0.028	1.556	0.045

Tested in the KITTI test set. The arrows indicate the preferred trend for each metric. Bold values mean the best score.

3) Distance Head Architecture: To improve distance estimation, we compare several head architectures for the ExtraHead: BaseConv, DeepConv, CBAM, Bottleneck, and DynamicBranching.

As shown in Table IV, DeepConv slightly improves mAP over BaseConv, suggesting deeper heads benefit detection. Bottleneck achieves the best MAE, MRE, and RMSLE, outperforming BaseConv on distance accuracy. While BaseConv shows a marginal RMSE advantage (3%), Bottleneck performs better in RMSLE, indicating improved performance on long-range predictions. Both Bottleneck and DynamicBranching are the most efficient, with 0.037s inference time.

Bottleneck's efficiency can be explained by its use of 1×1 convolutions, which compress and expand features for better representation, similar to its role in ResNet.

TABLE IV
PERFORMANCE AND INFERENCE TIME COMPARISON OF DIFFERENT DISTANCE HEAD ARCHITECTURES FOR THE DISTINANET MODEL

Architecture	mAP (↑)	MAE (↓)	MRE (↓)	RMSE (↓)	RMSLE (↓)	MIT (s)
BaseConv	0.806	0.678	0.031	**1.305**	0.056	0.038
DeepConv	**0.817**	0.680	0.031	1.337	0.057	0.040
CBAM	0.788	0.817	0.037	1.592	0.065	0.044
Bottleneck	0.803	**0.677**	**0.030**	1.346	**0.052**	**0.037**
DynamicBr.	0.801	0.740	0.034	1.425	0.061	**0.037**

Tested in the KITTI test set. Arrows indicate the preferred trend for each performance metric. Bold values mean the best score.

4) Distance Loss: This experiment compares loss functions for distance estimation: L1, L2, SmoothL1, Huber, and Log-Cosh. As shown in Table V, Huber Loss achieves the best overall performance, outperforming L1, L2, SmoothL1, and Log-Cosh by 4.8%, 9.2%, 3.4%, and 5.5%, respectively. SmoothL1 ranks second, consistent with its similarity to Huber.

Huber's blend of L1 and L2 properties offers robustness to outliers while maintaining sensitivity to small errors, making it suitable for stable and accurate distance regression.

TABLE V
COMPARISON OF DIFFERENT LOSS FUNCTION PERFORMANCE METRICS ON THE DISTINANET MODEL

Loss Function	mAP (↑)	MAE (↓)	MRE (↓)	RMSE (↓)	RMSLE (↓)
L1	0.806	0.678	0.031	1.305	0.056
L2	0.804	0.711	0.034	1.310	0.061
SmoothL1	0.800	0.668	0.031	1.237	**0.053**
Huber	**0.811**	**0.645**	**0.030**	**1.200**	**0.053**
Logcosh	0.808	0.683	0.032	1.656	0.056

Tested in the KITTI test set. Arrows indicate the preferred trend for each performance metric. Bold values mean the best score.

C. Qualitative Results

Fig. 6 shows qualitative results of DistinaNet and DistinaNet-BBoxExt on real-world images from our custom data set. Both models were trained on KITTI and evaluated on previously unseen data. The predictions are generally accurate, and we expect further improvements by incorporating images from cameras with varying intrinsic parameters to boost generalization.

Fig. 6. Qualitative results of the presented DistinaNet and DistinaNet-BBoxExt models in both KITTI test set and collected unseen real-world images. The values are presented in meters.

V. Conclusion

In conclusion, we propose two single-stage object-specific monocular distance estimation models: DistinaNet and DistinaNet-BBoxExt. Both achieve SOTA performance on KITTI and generalize well to real-world data collected via our custom annotation pipeline. Ablation studies show that adding distance estimation does not reduce detection accuracy and that using FPN layers 3–5 improves accuracy while reducing model size. The Bottleneck head, leveraging 1×1 convolutions, yields the best distance accuracy. Among loss functions, Huber Loss proves most effective due to its balance between robustness and sensitivity.

Acknowledgment

This work was supported by the Technology Innovation Program (RS-2025-02222776, RS-2024-00507520) funded by the Ministry of Trade, Industry and Energy (MOTIE, Korea).

References

[1] D. Rukhovich, D. Mouritzen, R. Kaestner, M. Rufli, and A. Velizhev, "Estimation of absolute scale in monocular slam using synthetic data," 2019.

[2] M. A. Haseeb, J. Guan, D. Ristić-Durrant, and A. Gräser, "Disnet: A novel method for distance estimation from monocular camera," 2018.

[3] A. Saxena, J. Schulte, A. Y. Ng *et al.*, "Depth estimation using monocular and stereo cues." in *IJCAI*, vol. 7, 2007, pp. 2197–2203.

[4] J. Janai, F. Güney, A. Behl, and A. Geiger, "Computer vision for autonomous vehicles: Problems, datasets and state of the art," 2021.

[5] S. Ghosh, J. Patrikar, B. Moon, M. M. Hamidi, and S. Scherer, "Airtrack: Onboard deep learning framework for long-range aircraft detection and tracking," 2023.

[6] J. Zhu, Y. Fang, H. Abu-Haimed, K.-C. Lien, D. Fu, and J. Gu, "Learning object-specific distance from a monocular image," 2019.

[7] M. Vajgl, P. Hurtik, and T. Nejezchleba, "Dist-yolo: Fast object detection with distance estimation," *Applied Sciences*, vol. 12, no. 3, 2022.

[8] D. Silva, N. Jourdan, and N. Gählert, "Long range object-level monocular depth estimation for uavs," 2023.

[9] M. Afshar, Z. Shirmohammadi, A. Ghahramani, A. Noorparvar, and A. Hemmatyar, "An efficient approach to monocular depth estimation for autonomous vehicle perception systems," *Sustainability*, vol. 15, p. 8897, 05 2023.

[10] Y.-C. Lai and Z.-Y. Huang, "Detection of a moving uav based on deep learning-based distance estimation," *Remote Sensing*, vol. 12, no. 18, 2020.

[11] M. Ahishali, M. Yamac, S. Kiranyaz, and M. Gabbouj, "Representation based regression for object distance estimation," *CoRR*, vol. abs/2106.14208, 2021.

[12] 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," 2020.

[13] R. Ranftl, K. Lasinger, D. Hafner, K. Schindler, and V. Koltun, "Towards robust monocular depth estimation: Mixing datasets for zero-shot cross-dataset transfer," *IEEE transactions on pattern analysis and machine intelligence*, vol. 44, no. 3, pp. 1623–1637, 2020.

[14] J. Li, X. Wang, Z. Wen, H. Dong, and J. Chen, "Hrdnet: A novel high-resolution depth estimation network," in *Proceedings of the 4th International Conference on Artificial Intelligence and Computer Engineering*, ser. ICAICE '23. New York, NY, USA: Association for Computing Machinery, 2024, p. 881–885.

[15] H. Liang, Z. Ma, and Q. Zhang, "Self-supervised object distance estimation using a monocular camera," *Sensors*, vol. 22, no. 8, 2022.

[16] Y. Zhang, , Y. Li, M. Zhao, and X. Yu, "A regional regression network for monocular object distance estimation," in *2020 IEEE International Conference on Multimedia Expo Workshops (ICMEW)*, 2020, pp. 1–6.

[17] M. Ochs, A. Kretz, and R. Mester, "Sdnet: Semantically guided depth estimation network," 2019.

[18] M. A. Reza, J. Kosecka, and P. David, "Farsight: Long-range depth estimation from outdoor images," in *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2018, pp. 4751–4757.

[19] H. Guan, C. Song, Z. Zhang, and T. Tan, "Monopoly: A practical monocular 3d object detector," *Pattern Recognition*, vol. 132, p. 108967, 2022.

[20] H. Yu and J. Oh, "Anchor distance for 3d multi-object distance estimation from 2d single shot," 2021.

[21] W. Song, Y. Sun, Q. Huang, and J. Cheok, "Side collision detection model for visually impaired using monocular object-specific distance estimation and multimodal real-world location calculation," Apr. 2024.

[22] S. Usmankhujaev, S. Baydadaev, and J. W. Kwon, "Accurate 3d to 2d object distance estimation from the mapped point cloud data," *Sensors*, vol. 23, no. 4, p. 2103, 2023.

[23] A. Geiger, P. Lenz, C. Stiller, and R. Urtasun, "Vision meets robotics: The kitti dataset," in *Proceedings of the International Conference on Pattern Recognition (ICPR)*, 2012.

[24] T.-Y. Lin, P. Goyal, R. Girshick, K. He, and P. Dollár, "Focal loss for dense object detection," in *Proceedings of the IEEE international conference on computer vision*, 2017, pp. 2980–2988.

[25] X. Zhang and Y. Demiris, "Visible and infrared image fusion using deep learning," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 45, no. 8, pp. 10 535–10 554, 2023.

[26] T. Miao, H. Zeng, W. Yang, B. Chu, F. Zou, W. Ren, and J. Chen, "An improved lightweight retinanet for ship detection in sar images," *IEEE Journal of Selected Topics in Applied Earth Observations and Remote Sensing*, vol. 15, pp. 4667–4679, 2022.

[27] X. Cheng and J. Yu, "Retinanet with difference channel attention and adaptively spatial feature fusion for steel surface defect detection," *IEEE Transactions on Instrumentation and Measurement*, vol. 70, pp. 1–11, 2020.

[28] P. F. Jaeger, S. A. Kohl, S. Bickelhaupt, F. Isensee, T. A. Kuder, H.-P. Schlemmer, and K. H. Maier-Hein, "Retina u-net: Embarrassingly simple exploitation of segmentation supervision for medical object detection," in *Machine Learning for Health Workshop*. PMLR, 2020, pp. 171–183.

[29] S. Woo, J. Park, J. Lee, and I. S. Kweon, "CBAM: convolutional block attention module," *CoRR*, vol. abs/1807.06521, 2018.

[30] C. Zheng, Y. Li, J. Li, N. Li, P. Fan, J. Sun, and P. Liu, "Dynamic convolution neural networks with both global and local attention for image classification," *Mathematics*, vol. 12, no. 12, 2024.

[31] K. Gokcesu and H. Gokcesu, "Generalized huber loss for robust learning and its efficient minimization for a robust statistics," 2021.

[32] C.-Y. Wang, I.-H. Yeh, and H.-Y. M. Liao, "Yolov9: Learning what you want to learn using programmable gradient information," *arXiv preprint arXiv:2402.13616*, 2024.

[33] Y. Davydov, W.-H. Chen, and Y.-C. Lin, "Supervised object-specific distance estimation from monocular images for autonomous driving," *Sensors*, vol. 22, no. 22, 2022.