AI-Driven Automatic Side Mirror Adjustment to Address Left-Side Blind Spots in Right Turns

Hyeonsu Kim, Jiwon Lee, Yeonsu Jeong, Seungjun Oh, Sumin Kim and Minseok Choi
Department of Electronic Engineering, Kyung Hee University, Yongin, South Korea
E-mails: nan111nan78@gmail.com, jiwonn612@khu.ac.kr, jys3049@khu.ac.kr,
sxungz@gmail.com, simoon06@khu.ac.kr, choims@khu.ac.kr

***Abstract*—The limited field of view in conventional side mirrors of vehicles poses serious safety risks, particularly during right turns or highway merges where the left-side rear area is often obscured. While digital side mirrors (DSM) offer partial improvements, they fall short in dynamically addressing blind spots in real time. This paper presents an intelligent side mirror control system that automatically adjusts the left-side mirror angle based on real-time driving context. To meet the dual demands of high accuracy and low latency, the system employs a lightweight convolutional neural network (CNN) with a BranchyNet-inspired multi-exit architecture. By enabling early exits under high-confidence predictions using entropy-based criteria, the model significantly reduces inference latency while maintaining reliable performance. Experiments on a custom real-world driving dataset demonstrate that the proposed method achieves up to 97.4% classification accuracy with an average inference time of 1.7 ms, making it suitable for embedded deployment in driver-assistance systems.**

***Index Terms*—Convolutional Neural Network, Multi-exit neural network, Advanced Driver Assistance Systems (ADAS), blind spot, real-time systems.**

I. Introduction

Ensuring comprehensive situational awareness is essential for safe driving, particularly during complex maneuvers such as highway merges and right turns at intersections. These scenarios often expose the left-rear zone of the vehicle to occlusions, creating blind spots that are difficult for drivers to perceive. Conventional side mirrors and even digital side mirrors (DSMs) are limited in their ability to fully eliminate these blind spots, especially under dynamic conditions. Inadequate visibility in such occluded regions has been identified as a key factor contributing to traffic accidents, particularly when drivers must make rapid decisions with limited information.

Recent work in autonomous driving has sought to characterize and mitigate blind spots through advanced perception and simulation techniques. For example, the authors of [1] used high-fidelity 3D simulations and Monte Carlo-based reference sensors to estimate sensor coverage and blind spot regions in autonomous vehicle designs. While their approach provides valuable insights into sensor limitations and layout optimization, it is intended for system design and evaluation rather than real-time support for human drivers. Similarly, Hubmann et al. [2] addressed safety in left-turn maneuvers at urban intersections using dynamic occupancy grid maps and multi-object tracking. Their system models unobservable regions and anticipates potential object movements to support trajectory planning. Although such methods are effective in autonomous systems, they require extensive sensor suites and high computational overhead, making them unsuitable for lightweight, real-time assistance in conventional vehicles.

Full-surround multi-object tracking (MOT) methods, such as those proposed by [3], have further improved environmental awareness by fusing data from camera and LiDAR sensors in a unified 3D coordinate space. While this enables robust tracking for autonomous navigation, it again depends on costly hardware configurations and is primarily aimed at high-level decision-making rather than direct driver support. Other research efforts have explored active physical interventions to enhance driver awareness. Kuwana et al. [4] proposed the Dynamic Angling Mirror System (DAMS) and its enhanced version (EDAMS), which adjust the mirror's yaw angle in response to nearby vehicles entering the blind spot. Their simulation-based study showed improved situational awareness and reduced collision risk, particularly for right-side blind spots during lane changes. However, these systems operate reactively, activating only after detecting nearby vehicles, and lack proactive prediction of high-risk scenarios based on driving context. Moreover, their scope was limited to right-side blind spots and did not incorporate intelligent perception mechanisms based on machine learning.

Despite these advancements, there remains a clear gap in developing lightweight, real-time mirror control systems that proactively respond to blind spot scenarios using onboard video streams and efficient neural models. Existing approaches either rely on complex sensor infrastructures or offer only limited reactive support without predictive situational awareness.

In this paper, we present an intelligent side mirror control system that proactively adjusts the left-side mirror to improve driver visibility during right turns and highway merges. Unlike existing systems designed for autonomous vehicles [2], [3]] or reactive mirror adjustment mechanisms [4], our approach supports human drivers using a lightweight, context-aware framework. The system utilizes onboard video streams and optional GPS data to detect high-risk scenarios, activating only under relevant driving conditions. This proactive control avoids unnecessary mirror movement and ensures that the system responds precisely to visibility-critical situations.

To achieve real-time responsiveness with high accuracy, we adopt a multi-exit CNN inspired by BranchyNet. The model enables early exits based on prediction confidence, significantly reducing latency without compromising reliability. A loss-weighting scheme during training further tunes the

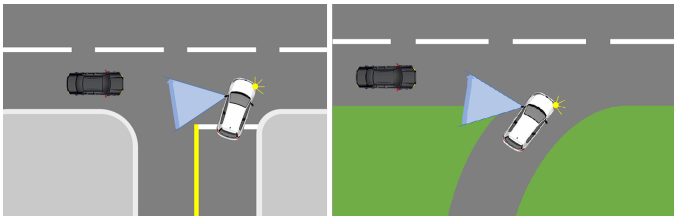

Fig. 1: Operational Conditions for the Automatic Downward-Tilting Side Mirror System

accuracy-latency trade-off, making the system adaptable to various deployment environments. Our experiments on a real-world driving dataset demonstrate that the proposed method achieves robust classification performance with minimal computational overhead, highlighting its feasibility for embedded deployment in Advanced Driver Assistance Systems (ADAS).

The remainder of this paper is organized as follows. Section II introduces the overall system architecture and outlines the problem formulation. Section III details the proposed multi-exit neural network structure, the entropy-based inference mechanism, and the training methodology. Section IV presents the experimental setup, dataset construction, and performance evaluation under various configurations. Finally, Section V concludes the paper and discusses potential directions for future work.

II. System Model

The proposed system targets a specific yet critical driving safety issue: the inability of conventional side mirrors to adequately reveal the left-side blind spot during right turns and highway merges. In such scenarios, the driver often lacks visibility into the diagonally rearward region, where nearby vehicles may approach. This issue is further compounded by the limited field of view of standard and even DSMs, which typically provide an angular coverage of only about 29 degrees, falling short of the 45 degrees or more required for safe merging or turning.

To address this challenge, we design an intelligent mirror control system that dynamically tilts the left-side mirror under specific driving conditions to expose the obscured area. This function is particularly vital when navigating intersections or highway entry ramps, where the inability to detect approaching vehicles can lead to serious collisions. A key aspect of the design is context-aware activation. The system avoids unnecessary mirror adjustments by evaluating the driving context rather than simply reacting to the use of a turn signal. For example, a signal-activated lane change on a straight road should not trigger the system, whereas a tight corner without signal use might still warrant mirror adjustment. This approach minimizes false activations and enhances system reliability.

To determine activation conditions, the system processes real-time input from a forward-facing camera (e.g., dashboard or black box camera), and optionally leverages GPS data for geographic context. A lightweight CNN model analyzes the

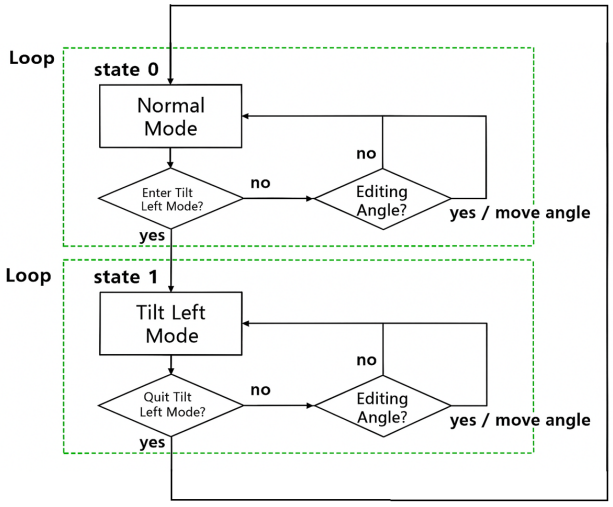

Fig. 2: State diagram of the process for setting the angle of the automatic left-hand side mirror

video frames to assess whether the current situation corresponds to predefined high-risk conditions, including:

- Approaching a right-turn intersection or highway merge zone
- Detectable vehicle motion indicating a turning or merging maneuver
- Limited visibility in the left-side diagonal field

The CNN outputs a binary classification indicating whether mirror adjustment is required. To enable low-latency response, the model adopts a BranchyNet-style architecture with multiple early-exit branches, allowing inference to terminate early in simple or high-confidence scenarios. Figure 1 illustrates the operational scenarios targeted by the system, and Figure 2 presents the control flow diagram of the mirror adjustment logic. In parallel, a GPS-based logic module refines system decisions by identifying predefined high-risk locations. The module cross-references current GPS coordinates with a stored database of known intersection or merge points. If the vehicle is outside of these zones, the system suppresses mirror adjustment, even if the CNN predicts activation, thereby preventing false positives in irrelevant contexts.

This dual-module architecture, which combines video-based scene classification with location-aware filtering, forms the foundation of the proposed system. It ensures that the mirror adjustment is both timely and contextually appropriate, enabling a responsive and computationally efficient solution to a long-standing visibility issue in human-driven vehicles.

III. Training of Wing Mirror Control Automation

To enable accurate and timely mirror control decisions, we trained a lightweight CNN to recognize driving situations that require left-side blind spot mitigation. This section introduces the dataset used for training, the design of the multi-exit neural network, and the training procedure.

Fig. 3: Samples of Dataset labeled as ‘0’ (Left) and ’1’ (Right)

A. Dataset

Since no public datasets directly address our problem, which detects driving contexts that warrant automatic mirror adjustment, we constructed a custom dataset tailored to this task. The data was collected from dashboard camera footage recorded during real-world driving across three metropolitan areas in South Korea: Incheon, Seoul, and Suwon. These locations were selected to reflect a wide range of road geometries, traffic patterns, and environmental conditions. To improve generalizability, we included various driving environments such as urban roads, intersections, and highways, as well as diverse conditions including day and night driving, and both clear and rainy weather. From the raw black box videos, individual frames were extracted based on the device’s native frame rate. We manually filtered the frames to retain only those containing intersections or highway merge points—situations in which blind spot awareness is especially important.

Each selected image was labeled using a binary scheme: frames corresponding to conditions that require mirror adjustment (e.g., approaching a right turn or merging scenario) were labeled as ‘0’, while others were labeled as ‘1’. Figure 3 shows example images for each class. In total, we collected 44,000 labeled images. For training, we used 20,000 images per class, and set aside 2,000 images per class for validation. Notably, the CNN model does not use GPS data directly. Instead, geographic context, such as whether the vehicle is near a known intersection, is handled separately by a GPS-based logic module during deployment. For this, we recorded GPS coordinates of key intersections and merge points in the Suwon area, which are used to suppress false activations outside relevant zones. This dataset serves as the foundation for training and evaluating the proposed model in real-world driving scenarios.

B. Multi-Exit CNN Architecture

To achieve a balance between inference accuracy and responsiveness, we adopt a multi-exit CNN architecture based on the BranchyNet framework [5]. This structure allows the network to produce predictions at multiple depths and exit early if the confidence is high, thereby reducing latency in straightforward cases without sacrificing performance in more complex scenes. Our implementation uses ResNet-101 as the backbone, with three exit branches placed after the first, third, and fifth convolutional blocks, respectively. Each branch includes a separate fully connected layer and softmax classifier. The final exit corresponds to the standard output

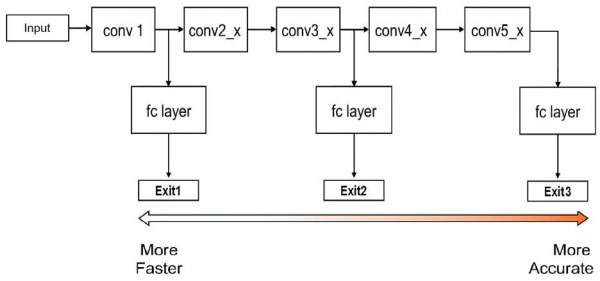

Fig. 4: The architecture of the BranchyNet-based ResNet-101

of the full ResNet-101 network. Figure 4 shows the overall architecture of the model.

During inference, the decision to exit at a given branch is based on the model’s confidence in its prediction. Specifically, we use the Shannon entropy of the softmax output at each exit to quantify uncertainty. Let $\hat{\mathbf{y}}_n = [\hat{y}_{n,1}, \hat{y}_{n,2}, \ldots, \hat{y}_{n,C}]$ be the softmax probability vector at the n-th exit, where $C = 2$ is the number of classes. The entropy at this exit is computed as:

$$H_n = -\sum_{c=1}^{C} \hat{y}_{n,c} \log \hat{y}_{n,c}. \tag{1}$$

Here, if the entropy H_n is below a predefined threshold T_n, the model exits at that branch and returns the corresponding class prediction. Otherwise, it continues to the next exit. This mechanism makes it possible to dynamically adapt the depth of inference based on input complexity.

The entropy threshold T_n acts as a tuning parameter for the speed–accuracy trade-off:

- A lower threshold requires higher confidence to exit early, resulting in deeper inference and potentially higher accuracy at the cost of increased latency.
- A higher threshold allows earlier exits, reducing average latency but with the risk of premature decisions in ambiguous scenes.

In this study, we set $T_n = 0.025$ for all exits, following recommendations from prior work [5]. We further analyze the effect of different thresholds in Section IV.

This early-exit mechanism is particularly well-suited for real-time applications like ADAS, where quick, confident decisions are preferred in typical situations, while deeper analysis is reserved for edge cases. Algorithm 1 outlines the entropy-based inference procedure used in our system.

C. Training Method

The goal of training is to ensure that all exit branches in the network produce reliable predictions, enabling the system to adaptively balance accuracy and latency. To this end, we define a composite loss function that combines the cross-entropy losses from each exit, allowing the model to learn representations useful at multiple depths.

Let $\hat{\mathbf{y}}_n = [\hat{y}_{n,1}, \hat{y}_{n,2}]$ denote the predicted class probabilities at exit $n \in \{1, 2, 3\}$, and let $\mathbf{y} = [y_1, y_2]$ denote the one-hot

Algorithm 1 BranchyNet Inference Algorithm

```
procedure BranchyNet Inference(x, T)
    for n = 1 ... N do
        z = f_exit_n(x)
        ŷ = softmax(z)
        e ← entropy(ŷ)
        if e < T_n then
            return arg max ŷ
        end if
    end for
    return arg max ŷ
end procedure
```

encoded ground truth label. The cross-entropy loss at exit n is defined as:

$$\mathcal{L}_n(\hat{\mathbf{y}}_n, \mathbf{y}) = -\sum_{c=1}^{2} y_c \log \hat{y}_{n,c}. \tag{2}$$

To jointly train the entire network, we aggregate the losses from all exits and generate the total training loss by taking the weighted sum of the individual exit losses, as given by

$$\mathcal{L}_{\text{total}} = \sum_{n=1}^{N} \gamma_n \mathcal{L}_n(\hat{\mathbf{y}}_n, \mathbf{y}), \tag{3}$$

where $\gamma_n \in [0, 1]$ is the weight assigned to the n-th exit, satisfying $\sum_{n=1}^{N} \gamma_n = 1$. This weighting allows fine-grained control over the model's behavior. For instance, giving more weight to early exits encourages the network to optimize for faster predictions, which is beneficial in latency-critical scenarios. Conversely, emphasizing deeper exits improves final accuracy at the cost of increased computation.

During backpropagation, gradients from each loss component are propagated through the shared feature extractor up to their respective exit points. This setup ensures that all branches contribute to training while allowing earlier layers to receive stronger gradient signals, which helps stabilize learning and reduces the risk of vanishing gradients. Additionally, this form of joint optimization acts as a form of regularization, improving generalization by aligning predictions across multiple abstraction levels [5]. We train the model using Stochastic Gradient Descent (SGD) with momentum. Table I summarizes the training hyperparameters. All training runs used a batch size of 64 and were conducted for 100 epochs. The learning rate was set to 0.01, with a momentum of 0.9 and a weight decay of 0.0001.

IV. Experimental Results

A. Experimental Environment

To evaluate the effectiveness of the proposed multi-exit CNN architecture, we conducted extensive training and validation using the custom dataset described in Section III-A. To ensure a fair evaluation, the test set was constructed from driving scenes that were entirely separate from the training and validation data.

TABLE I: Parameters and Environment for Training

Parameter	
Epoch	100
Batch Size	64
Momentum (for SGD optimizer)	0.9
Weight Decay (for SGD optimizer)	0.0001
Learning Rate	0.01

TABLE II: Accuracy and latency per exit

Exit	Acc. (%)	Latency (ms)	Usage (%)
Exit 1	93.750	0.68	64.1
Exit 2	96.438	2.5	22.1
Exit 3	98.656	4.72	13.8
Total	97.41	1.7	–

All experiments were performed on a high-performance desktop equipped with an NVIDIA GeForce RTX 3090 GPU (24 GB VRAM). The software environment consisted of Ubuntu 22.04 LTS, Python 3.9, and PyTorch 2.2. CUDA 11.8 and cuDNN 8.9 were used to accelerate GPU computation. The dataset was preprocessed by extracting frames from driving videos, resizing them to a consistent resolution, and balancing the number of samples across the two class labels. All annotations were performed manually and carefully verified for consistency. To ensure the reliability of results, each experiment was repeated three times. Performance metrics including classification accuracy and inference latency were averaged across runs to account for variance. This evaluation setup provides a realistic estimate of the model's performance under deployment conditions and reflects the computational demands of real-time operation in embedded systems.

B. Performance Evaluation

We evaluated the trained model based on two primary metrics: classification accuracy and inference latency. Special attention was given to the effectiveness of the entropy-based early-exit mechanism. Using a fixed entropy threshold of $T_n = 0.025$, we measured the performance at each exit branch individually. As shown in Table II, the accuracy increased with the depth of the network. The first exit achieved 93.75% accuracy with an average inference time of 0.68 milliseconds. The second exit improved accuracy to 96.44% with 2.5 milliseconds of latency, while the final exit reached 98.66% accuracy at a latency of 4.72 milliseconds.

When adaptive inference was applied using the entropy-based exit strategy, the system dynamically determined the appropriate exit point based on prediction confidence. Under the fixed threshold, the overall classification accuracy reached 97.41%, with the average inference time reduced to 1.7 milliseconds. This latency reduction was primarily driven by the frequent use of the earliest exit, which was selected in 64.1% of cases. The second and third exits were used in 22.1% and

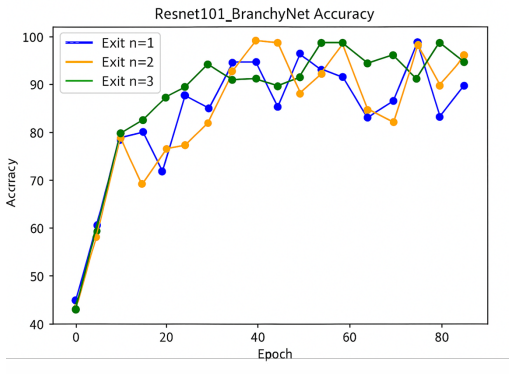

Fig. 5: Results from training with a weighted sum of $\gamma_1 = 0.4, \gamma_2 = 0.3, \gamma_3 = 0.3$

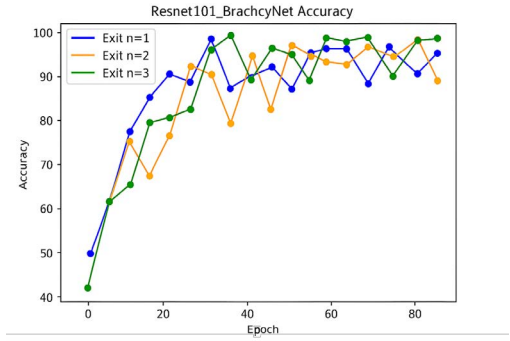

Fig. 6: Results from training with a weighted sum of $\gamma_1 = 0.5, \gamma_2 = 0.3, \gamma_3 = 0.2$

13.8% of cases, respectively. These results demonstrate that the proposed early-exit mechanism effectively maintains high prediction accuracy while significantly lowering computational cost. The ability to exit early in straightforward scenarios allows the system to respond quickly without compromising reliability, making it suitable for real-time use in driver-assistance applications where both speed and accuracy are critical.

C. Impact of Hyperparameters

To investigate how training-time loss weighting affects the model's behavior, we experimented with three different exit-wise weighting schemes: uniform, moderately front-loaded, and strongly front-loaded configurations. The results indicate that increasing the weight assigned to earlier exits generally improves their classification accuracy, as the model is more strongly encouraged to optimize performance at shallow layers. However, this comes at the cost of slightly reduced accuracy in the deeper exits, suggesting a trade-off in representational focus across the network.

For instance, when using a strongly front-loaded configuration with weights $\gamma_1 = 0.5$, $\gamma_2 = 0.3$, $\gamma_3 = 0.2$, the first exit achieved the highest accuracy among all configurations, but the final exit showed a minor drop in performance. This behavior reflects the effect of stronger gradient signals at the early stages of the network, which helps stabilize training for shallow exits while limiting the learning capacity of deeper branches. On the other hand, the uniform weighting scheme ($\gamma_1 = \gamma_2 = \gamma_3 = 0.333$) yielded the highest overall accuracy across all exits but resulted in a higher average inference time due to more frequent use of the later exits.

When tested under the entropy-based adaptive inference policy with a fixed threshold of $T_n = 0.025$, models trained with more front-loaded weights tended to favor earlier exits more confidently. As a result, these models showed lower average inference latency while still maintaining reasonable overall accuracy. This outcome demonstrates that loss weighting is a practical tool for shaping the model's runtime behavior. Applications requiring low-latency decision-making can benefit from assigning higher weights to earlier exits during training, whereas use cases demanding maximum accuracy may prefer uniform or deeper-weighted configurations.

TABLE III: Performance comparison with different γ weights

γ	Exit	Acc. (%)	Latency (ms)
$\gamma_1 = 0.333$	Exit 1	93.750	0.68
$\gamma_2 = 0.333$	Exit 2	96.438	2.5
$\gamma_3 = 0.333$	Exit 3	98.656	4.72
T_n = 0.025	**Overall**	**97.41**	**1.7**
$\gamma_1 = 0.4$	Exit 1	94.340	0.68
$\gamma_2 = 0.3$	Exit 2	96.568	2.5
$\gamma_3 = 0.3$	Exit 3	97.104	4.72
T_n = 0.025	**Overall**	**96.98**	**1.5**
$\gamma_1 = 0.5$	Exit 1	95.134	0.68
$\gamma_2 = 0.3$	Exit 2	95.198	2.5
$\gamma_3 = 0.2$	Exit 3	96.882	4.72
T_n = 0.025	**Overall**	**95.2**	**0.88**

TABLE IV: Performance comparison with different T_n thresholds

T_n	Exit	Acc. (%)	Usage (%)
0.01	Exit 1	93.750	53.4
	Exit 2	96.438	32.1
	Exit 3	98.656	14.5
$\gamma = 0.333$	**Overall**	**98.45**	-
0.025	Exit 1	93.750	64.1
	Exit 2	96.438	22.1
	Exit 3	98.656	13.8
$\gamma = 0.333$	**Overall**	**97.41**	-
0.05	Exit 1	93.750	85.2
	Exit 2	96.438	10.9
	Exit 3	98.656	3.9
$\gamma = 0.333$	**Overall**	**94.24**	-

We further analyzed the effect of varying the entropy threshold T_n itself, which directly influences the model's tendency to exit early. A more lenient threshold of 0.05 led to more aggressive early termination, with over 85% of predictions completed at the first exit. While this greatly reduced average latency, it also slightly decreased overall accuracy. In contrast, a stricter threshold of 0.01 required higher confidence for early exits, shifting more predictions to the deeper branches and achieving higher overall accuracy at the cost of increased inference time.

These results highlight the flexibility of the proposed framework, where training-time loss weights and inference-time entropy thresholds offer complementary ways to balance speed and accuracy. Depending on system constraints and application requirements, the model can be tuned to favor responsiveness, precision, or a desired balance of both.

V. Conclusion

This paper presented an intelligent side mirror control system designed to proactively address left-side blind spots during right turns and highway merges. To achieve both fast and reliable decision-making, we employed a multi-exit CNN architecture inspired by BranchyNet, allowing early exits based on entropy thresholds. The model was trained using a custom real-world driving dataset and optimized via a weighted loss function that balances accuracy and inference latency. Experimental results demonstrated that the proposed approach maintains high classification accuracy while significantly reducing average inference time. By adjusting the loss weights across exits and tuning the entropy threshold during inference, the system can be tailored to different performance requirements, making it well-suited for real-time deployment in embedded driver-assistance systems.

Acknowledgment

This work was supported in part by Institute of Information & communications Technology Planning & Evaluation (IITP) under YKCS Open RAN Global Collaboration Center (IITP-2025-RS-2024-00434743) grant funded by the Korea government (MSIT), and in part by the National Research Foundation of Korea (NRF) grant funded by the Korea government (MSIT) (RS-2025-00521295).

References

[1] D. Rohr, D. Isele, and J. Bohg, "Can you see me now? blind spot estimation for autonomous vehicles using scenario-based simulation with random reference sensors," in *IEEE Intelligent Vehicles Symposium (IV)*, 2021, pp. 428–434.

[2] C. Hubmann, M. Becker, M. Althoff, and C. Stiller, "Entering crossroads with blind corners: A safe strategy for autonomous vehicles," in *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2017, pp. 1724–1731.

[3] A. Rangesh and M. M. Trivedi, "No blind spots: Full-surround multi-object tracking for autonomous vehicles using cameras and lidars," *IEEE Transactions on Intelligent Vehicles*, vol. 4, no. 4, pp. 588–602, 2019.

[4] J. Kuwana, M. Itoh, and T. Inagaki, "Dynamic side-view mirror: Assisting situation awareness in blind spots," in *2013 IEEE Intelligent Vehicles Symposium (IV)*, 2013, pp. 455–460.

[5] S. Teerapittayanon, B. McDanel, and H. Kung, "Branchynet: Fast inference via early exiting from deep neural networks," 2017.