CAMVA: An Extension Architecture of CNN Accelerators for Multi-View Acceleration

Junjie Chen, *Student Member, IEEE*, Yan Ding, *Member, IEEE*, Chubo Liu, *Member, IEEE*, and Keqin Li, *Fellow, IEEE*

***Abstract*—Autopilot vehicles integrate additional views to capture comprehensive feature information, enhancing target detection accuracy. However, this integration imposes a higher computational burden on CNN accelerators in autopilot systems, potentially increasing the response latency of autopilot systems. It is a challenge for safety-critical autopilot systems. Traditionally, researchers have addressed this challenge by employing complex designs and processes to enhance the arithmetic capabilities of chips. In contrast, the paper proposes a simple technology that is an extension architecture of CNN accelerators for multi-view acceleration (CAMVA). The extension architecture utilizes the characteristic of multi-view approximation to enhance the sparsity of input features and dynamically reuse the approximated feature extraction results, accelerating CNN accelerators. This paper uses multi-view autonomous driving datasets (KITTI and nuScenes) to create two tasks, and evaluates the impact of CAMVA on 2D and 3D object detection networks by simulating their data flows. Results of 2D experiments show that for 2D object detection, the accuracy of the KITTI task decreases by 2.29%∼4.26%, while that of the nuScenes task decreases by 0.37%∼1.11%. For 3D object detection, the accuracy of the KITTI task decreases by 1.97%∼-0.22%. Then, the paper utilizes the pruning operation of CAMVA to create two subtasks, which are subsets of the KITTI with sparsity of 9.6% and 19.8%, respectively. It evaluates the performance, energy consumption, and RTL of CAMVA at the hardware architecture level based on the two subtasks. The results show that CAMVA enhances the performance of CNN accelerator by 1.04∼1.08× and reduces energy consumption by 4.23%∼8.01% in the sparsity 9.6% subtask; additionally, it improves performance by 1.13∼1.34× and decreases energy consumption by 14.13%∼25.97% in the sparsity 19.8% subtask. CAMVA increases CNN accelerator's area by a mere 0.75%.**

Received 21 February 2025; revised 11 July 2025; accepted 16 October 2025. Date of publication 23 October 2025; date of current version 11 December 2025. This work was supported in part by NSFC under Grant 62572180, in part by the Science and Technology Innovation Program of Hunan Province under Grant 2024RC3076, and in part by the Fundamental Research Funds for the Central Universities. Recommended for acceptance by R. Wang. *(Corresponding author: Yan Ding.)*

Junjie Chen, Yan Ding, and Chubo Liu are with the College of Computer Science and Electronic Engineering, Hunan University, Hunan 410082, China, and also with the National Supercomputing Center in Changsha, Hunan 410082, China (e-mail: chenjunjie2020@hnu.edu.cn; ding@hnu.edu.cn; liuchubo@hnu.edu.cn).

Keqin Li is with the College of Computer Science and Electronic Engineering, Hunan University, Hunan 410082, China, and also with the National Supercomputing Center in Changsha, Hunan 410082, China, and also with the Department of Computer Science, State University of New York, New Paltz, NY 12561 USA (e-mail: lik@newpaltz.edu).

Digital Object Identifier 10.1109/TC.2025.3624213

***Index Terms*—Autopilot, multi-view, approximate, pruning, restoration, accelerating, CNN accelerator.**

I. Introduction

AUTOPILOT technology relies on perception systems equipped with a target detection module to detect and recognize objects in the surrounding environment accurately. In autopilot companies, such as Tesla [1], Mobileye [2], and Foresight [3], the target detection module comprises image feature detection and image target detection. Image feature detection extracts key features to reinforce the functioning of autopilot systems, such as multisensor data fusion [4], autonomous navigation [5]. Image target detection utilizes convolutional neural networks (CNNs) to implement perception tasks, e.g. target identification, road recognition, and decision making, ensuring the safety and high efficiency of autopilot vehicles [6].

Target detection modules, as an essential component of autopilot perception systems, are transitioning from single-view detection to multi-view detection to achieve superior perception performance. Multi-view detection necessitates the integration of multiple cameras capturing simultaneous images from diverse perspectives, thereby expanding the field of view and enhancing detection accuracy. The increased camera numbers result in a larger input data volume per unit of time, imposing a significant computational burden on CNN accelerators in autopilot vehicles and potentially increasing the response latency of autopilot systems. The presence of overlapping regions in these data exhibits the characteristic of convolutional feature approximation [7] and local feature approximation [8], leading to a large number of redundant features across views. However, most of the existing efficient techniques only utilize intra-view redundancy features for acceleration, without considering inter-view relationships, and cannot effectively address the latency challenges posed by multi-view scenarios in autopilot systems.

Traditionally, researchers have endeavored to tackle this issue of declining real-time performance of accelerators by enhancing the computational power of CNN accelerators through complex designs and processes, which are costly and not conducive to rapid development. In contrast, the paper proposes an extension architecture of CNN accelerators for multi-view acceleration (CAMVA). It enhances the inference performance of CNN accelerators [9], [10], [11] in multi-view scenes by pruning approximate redundant input features and reusing results in multi-view scenes without incurring additional overheads or

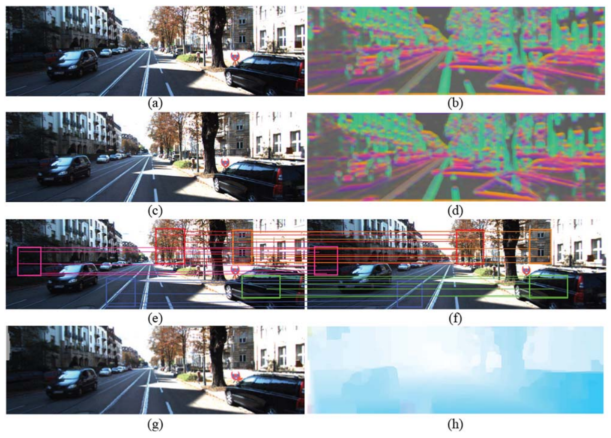


Fig. 1. The approximate relationships in multi-view.

requiring more expensive and power-consuming alternatives. CNN accelerators are able to store such input features in a suitable format [10] to facilitate the elimination of zero-valued inputs. Additionally, the current mainstream multi-view matching methods are designed for dual-view and may not be suitable for multi-view. Therefore, the paper proposes a novel Multi-View Dynamic Matching Algorithm (MVDMA) to accurately match approximation regions in multi-view tasks with excellent real-time performance.

The scale-invariant feature transform (SIFT) flow [8] is employed to demonstrate the approximation in multi-view. Specifically, the SIFT flow is a technique that aligns an image with other images and visualizes differences. Fig. 1 depicts the approximation relationship [12] between the two perspectives in the KITTI dataset. Fig. 1(a) and 1(b) display the left view captured by the binocular camera and its corresponding visualized SIFT descriptors [13], while (c) and (d) exhibit the right view and its corresponding visualized SIFT descriptors. The visually approximate regions between views are depicted with similar colors in their visualized SIFT descriptors. Fig. 1(g) shows (c) warped onto (a). Fig. 1(h) presents the estimated SIFT flow field, which reflects the degree of dissimilarity between views. Consequently, the variations in the SIFT flow field are stable, and differences between views are negligible.

Our contributions to this work include the following:

1) We propose CAMVA, which accelerates inference and saves energy by skipping unnecessary data access and computation, reusing computed results, without compromising detection accuracy. It benefits the reduction of reaction latency in the autopilot systems.
2) We propose a multi-view dynamic matching algorithm, which utilizes SIFT features to match approximate regions in multi-view and provides metadata to CAMVA.
3) We evaluate the validity of reusing approximate results in multi-view, showcase the impact of CAMVA on the performance of the CNN accelerator, and demonstrate the advantages of CAMVA in energy consumption and real-time performance for end-to-end applications.

The rest of this paper is organized as follows. Section II presents background and motivation. Section III discusses the algorithms for CAMVA, including the SIFT algorithm, multi-view dynamic matching algorithm, pruning strategy, dynamic feature restoration strategy, and error correction strategy. Section IV discusses the architecture for CAMVA in the CNN accelerator, data flow, and feature restoration. Section V describes the experimental methods. Section VI describes the experimental results. Section VII concludes the paper.

II. Background and Motivation

The paper is based on two research fields: multi-view alignment and convolutional neural network (CNN) accelerators.

A. Multi-View Alignment

Multi-view is a crucial research direction in computer vision, robotics, and autopilot. It utilizes multiple viewpoints or sensors to capture objects or scenes information and has a broad range of applications, such as image alignment [14].

Image alignment is a fundamental technique and a central topic in computer vision. It is commonly employed to investigate various tasks, including image-stitching [15], stereo matching [16]. In the past, traditional approaches for matching views relied on techniques such as phase [17], filter banks [18], and gradient analysis [19]. However, these techniques often yield unreliable results when confronted with rapidly changing images. Nowadays, researchers widely realize the significance of intermediate-level representations in image alignment. These include SIFT [13], shape context [20], and histograms of oriented gradients [21], which capture significant differences between images and have proven effective in various applications. The two types of intermediate-level representations are dense and sparse representations, respectively. Dense representations establish correspondences at the pixel level, such as optical flow for motion analysis and differential fields for stereo vision. Sparse representations establish correspondences at the feature level using techniques like Harris [22] and SIFT [8], effectively identifying local or global image approximations.

The SIFT algorithm extracts scale, rotation, and illumination invariant features extensively used in diverse autopilot systems. Therefore, the paper utilizes SIFT features as the matching feature for MVDMA. Considering the high computation and energy consumption of the SIFT algorithm, autopilot vehicles employ SIFT accelerators [23], [24] to reduce costs in extracting these features from input images.

B. CNN Accelerator

In autopilot systems, the strategy of using multiple cameras not only helps compensate for data degradation and feature insufficiency problems that occur during target detection but also effectively tackles common challenges in the real world, such as target occlusion and depth perception. However, the current multi-view technique has a fundamental drawback in the form of duplicate approximation regions between multiple views. These regions are redundantly processed multiple times during image rendering and target detection, resulting in a significant computational burden on both the image rendering module and target detection module of autopilot vehicles. Furthermore, as the number of views increases, the time required for image rendering and target detection increases linearly, leading to higher reaction latency for autopilot vehicles with strict safety requirements. Many researchers have investigated skipping these overlapping regions to avoid repeatedly rendering them during multi-view rendering. Inspired by this research, this paper focuses on approximating computations for the target detection module in multi-view autopilot systems to effectively skip redundant computation and improve system efficiency.

Target detection neural networks typically employ approximate computing techniques, such as quantization methods, weight approximation, and computational reuse, to improve inference performance. However, due to limitations in computational power and energy consumption of embedded systems, researchers often design specific CNN accelerators for different approximate computing techniques. Based on the good fault-tolerance property of CNN [25], quantization accelerators use low-bit data for approximate computation to enhance the performance of inference and training while reducing storage and transmission costs. For example, Jung et al. [26] proposed a quantizer that accelerates computation and compresses the model with an accuracy comparable to that of a full-precision network. Sparse accelerators utilize skipping invalid computation techniques to improve inference performance and reduce energy consumption. The skipping operations can be divided into two types: feature skipping which compresses zero values in input features to reduce data accesses and computations; and weight skipping which avoids loading parameters with zero weights into processing element (PE) arrays to reduce data accesses and computations. For example, Cnvlutin [9] utilizes sparsity in the input feature to improve the performance of accelerators. Reuse accelerators focus on data reuse to improve CNN inference performance and reduce energy consumption by reusing features or sharing weights among PEs. Eyeriss [27] achieves efficient convolutional computation by reusing features across multiple layers.

The techniques effectively enhance CNN inference performance and save energy. However, further exploration of the multi-view potential for improving CNN inference performance is necessary to address latency challenges in multi-view applications. This paper deeply explores redundancy characteristics in multi-view target detection using approximate computing techniques and proposes an extension architecture that is compatible with high-efficiency techniques and applicable to multi-view scenarios without affecting the operation mode of existing CNN accelerators. CAMVA is not a standalone accelerator but an extension architecture of CNN accelerators that can accelerate ones supporting one-sided sparse input, reducing computational and data access overheads.

III. Algorithm for CAMVA

This section describes the algorithm for providing metadata to CAMVA based on the execution flow. First, the SIFT algorithm extracts view features. Then, the meta_process_unit (MPU) uses the Multi-View Dynamic Matching Algorithm (MVDMA) to match views, construct macroblocks, and prune macroblock features. Finally, CAMVA restores features with error correction using metadata to accelerate CNN inference.

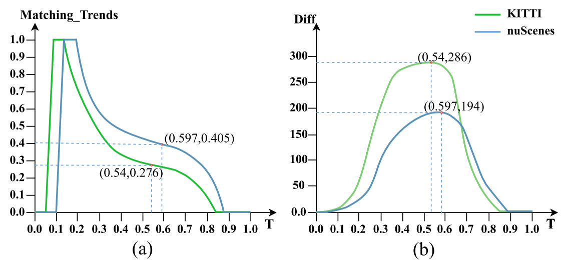


Fig. 2. The correlation of T and SIFT features in the autopilot datasets.

A. SIFT Algorithm

The scale-invariant feature transfor (SIFT) algorithm extracts SIFT features, which exhibit a notable degree of robustness against variations in viewpoint, affine transformations, and noise, rendering the algorithm one of the most dependable techniques for extracting image features [28].

After SIFT feature extraction, the SIFT features are matched using Lowe's ratio test [13], with the hyperparameter T being set. T denotes the Euclidean distance ratio between the nearest and the next closest matched key points [29], which determines the matching quality. To investigate the correlation between T and the autopilot datasets (KITTI and nuScenes), RANSAC [30] is employed to distinguish between the correct and error-matched features, using $Num_{correct}$ and Num_{error} to represent their respective quantities. Eq. (1):

$$Matching\_Trends = \frac{Num_{correct} - Num_{error}}{total\_matches}, \quad (1)$$

represents the normalized discrepancy between the correct and error-matched features, which assesses the relative matching accuracy in a set of features. Fig. 2(a) depicts the matching trends for different values of T on KITTI and nuScenes. Fig. 2(b) displays the difference between the correct and error-matched features, denoted as Diff, for various values of T in KITTI and nuScenes. The experimental results show that both datasets have higher correct matching rates and more matching features for $T=$ 0.54 and 0.597, respectively.

SIFT features exhibit remarkable capabilities in feature representation, but their computational overhead cannot be overlooked. Therefore, the autopilot perception system employs SIFT accelerators to reduce the computational overhead of extracting SIFT features, with the features then stored in memory [31], [32].

B. Multi-View Dynamic Matching Algorithm

The Multi-View Dynamic Matching Algorithm (MVDMA) based on SIFT features consists of two components: matching multi-view and constructing approximate regions.

Algorithm 1 Multi-view Matching Features Algorithm

```
inpput: MMVSet = {MV_{1,2},...,MV_{n-1,n}}
output: multiview_match_features
1:  Initialize empty match list $m\_list \leftarrow \emptyset$
2:  for each $MV_v \in$ MMVSet do
3:      for each $fp_k$: $(fp_{v,k}, fp_{v+1,k}) \in MV_v$ do
4:          $matched \leftarrow$ False
5:          for each existing set $S \in m\_list$ do
6:              if $fp_{v,k} \in S$ or $fp_{v+1,k} \in S$ then
7:                  $S \leftarrow S \cup \{fp_{v,k}, fp_{v+1,k}\}$
8:                  $matched \leftarrow$ True
9:          if not $matched$ then
10:             $m\_list \leftarrow m\_list \cup \{\{fp_{v,k}, fp_{v+1,k}\}\}$
11:         for each $MV_w \in$ MMVSet and $MV_w != MV_v$ do
12:             for each $(fp'_{v,k}, fp'_{v+1,k}) \in MV_w$ do
13:                 for each $S \in m\_list$ do
14:                     if $fp'_{v,k} \in S$ or $fp'_{v+1,k} \in S$ then
15:                         $S \leftarrow S \cup \{fp'_{v,k}, fp'_{v+1,k}\}$
16: Construct multiview_matches_features from $m\_list$
17: return multiview_matches_features
```

1) Matching Multi-View: Multi-view matching refers to matching features across two or more views. The paper proposes a multi-view matching algorithm based on the SIFT feature to address the issue. Algorithm 1 describes the multi-view matching process.

Firstly, Algorithm 1 loads SIFT features of each view from memory and builds the matched features groups in multi-view, denoted as $MMVSet = \{\{MV_{1,2}, MV_{2,3}, \ldots, MV_{n-1,n}\}, n$ = views number$\}$, where $MV_{i,i+1}$ =$\{\{fp_0, fp_1, \ldots, fp_m\}$, m=features number$\}$ represents the matched features group between $view_i$ and $view_{i+1}$.

fp_m= $(fp_{i,m}$: $fp_{i+1,m})$ represents the matched relationship between the feature m in $view_i$ and its matched feature m in $view_{i+1}$. $fp_{i,m}$ contains 128-dimensional features and the coordinates (x, y) of m in $view_i$.

Secondly, Algorithm 1 establishes the matching relationship between views and features by utilizing matched features within $MMVSet$ to generate a table (multiview_match_features (mmf)). mmf represents matched features groups in multi-view, categorized as complete or incomplete. A complete group means each feature has matches in every view, while an incomplete group means each feature only has matches in some views. Fig. 3 shows an example of matched features in three views ($view_a$, $view_b$, and $view_c$). Table I depicts the matched relationship between views and features in mmf. fp_6 represents a group of complete matched features (underlined), while fp_i (i=1, 2, 3, 4, 5) represents a group of incomplete matched features.

2) Constructing Approximate Regions: The discreteness of SIFT features presents a challenge for CNN in extracting valuable information. To address this, the paper proposes an approximate region construction method inspired by image stitching

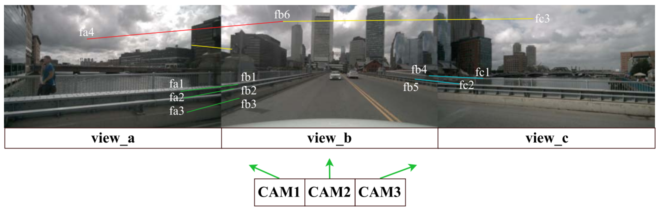


Fig. 3. The example of matched features in three views.

TABLE I
THE EXAMPLE OF MULTIVIEW_MATCH_FEATURES

	fp_1	fp_2	fp_3	fp_4	fp_5	fp_6
$view_a$	fa_1	fa_2	fa_3	-	-	$\underline{fa_4}$
$view_b$	fb_1	fb_2	fb_3	fb_4	fb_5	$\underline{fb_6}$
$view_c$	-	-	-	fc_1	fc_2	$\underline{fc_3}$

[15], which involves clustering and transforming the SIFT features into macroblock (MB) features.

Clustering The paper applies the Density-Based Spatial Clustering of Applications with Noise (DBSCAN) algorithm [33] to cluster features in each view, overcoming the challenge of extracting valuable information from the discreteness of SIFT features. The algorithm is an efficient unsupervised machine learning technique that assumes close spatial connectivity among samples of the same category [34]. It has a time complexity of $\mathcal{O}$ (nlogn) and does not require pre-setting the number of clusters, making it suitable for dense datasets of any shape. The clustered objects within a view consist of SIFT features in the same category. These clusters exhibit consistent matches with their corresponding SIFT features in multi-view. This aligns with the concept proposed in this paper for constructing approximate regions based on SIFT features. For example, the clustering in Table I results in $\{fp_1, fp_2, fp_3\}$, $\{fp_4, fp_5\}$ and $\{fp_6\}$. The performance of DBSCAN is influenced by two crucial parameters: $minPts$ and $epsilon$, which represent the minimum number of data points required for a core point in its neighborhood and the radius size used to define a data point's neighborhood, respectively. Appropriate parameters are chosen based on the dataset, and this paper utilizes methods [34], [35] to determine these parameters.

Constructing The clustered objects constructed by the clustered SIFT features are approximately [7], and the paper uses bounding_boxes, i.e., MBs, to mask the features in the clustered. This conversion transforms the matched relationship of features into that of MBs. The location of MB is determined by clustering features (FP^c=$\{fp^c_{v,1}, fp^c_{v,2}, \ldots, fp^c_{v,m}$, c=$\{1, 2, \ldots, n\}$, m=features number, n=clusters number), where c denotes the index of the clustered object, and the minimum and maximum x and y coordinates are selected to calculate its width ($w = x_{max} - x_{min}$), height ($h = y_{max} - y_{min}$), and center coordinates ($\frac{1}{2}(x_{max} + x_{min}), \frac{1}{2}(y_{max} + y_{min})$).

C. Pruning of Redundant Macroblocks Strategy

This subsection proposes a redundant macroblocks pruning strategy to enhance the sparsity of multi-view inputs.

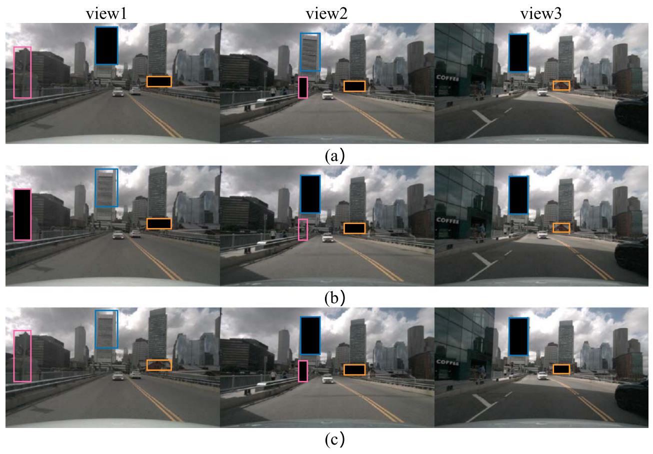


Fig. 4. The results of three different pruning strategy.

The set of matched macroblocks retains one specific macroblock, referred to as the retention macroblock (RTMB), while the remaining macroblocks are designated as redundant macroblocks (RDMBs) and undergo feature pruning. The pruning strategy is crucial in the study as it can conserve more resources. Therefore, the study evaluates three pruning strategies: Round-Robin strategy (RR_S), Load-Balancing strategy (LB_S), and Maximum-Approximate-Region-Retention strategy (MARR_S) using FasterRCNN on the KITTI dataset.

RR_S allocates RTMB and RDMB in each group of matched MBs based on the index of view using a Round-Robin approach, as illustrated in Fig. 4(a). This strategy guarantees that the number of RTMBs is evenly distributed across all views. LB_S aims to minimize the discrepancy in the total area of pruned RDMBs across different views, as illustrated in Fig. 4(b). This strategy ensures equitable computational cost distribution among views, optimizing resource utilization. MARR_S selects the matched MB with the largest area in a set of matched MBs as RTMB, as illustrated in Fig. 4(c). This strategy preserves the most features within each matched MB group, facilitating feature extraction and restoration.

The paper uses the raw results as the baseline (normalized: 1, mAP: 70.41%) to assess the impact of the three strategies on the performance and accuracy of FasterRCNN, respectively. Compared with the baseline, the normalized performance and accuracy of the three strategies are RR_S (0.91, 68.11%), LB_S (0.96, 67.59%), and MARR_S (1.26, 62.22%). MARR_S demonstrates a significant performance improvement but lower accuracy, whereas RR_S and LB_S exhibit good accuracy but minimal performance improvements compared to the raw results. Two factors contribute to this phenomenon. Firstly, in RR_S and LB_S, the restoration operation for partially matched MBs is triggered prematurely, hindering the computation reduction. Secondly, the larger area of RDMBs compared to RTMBs leads to additional interpolation computations during restoration, resulting in increased computational overhead. In contrast, MARR_S avoids restoring any RDMB feature extraction results in advance, and the RTMB feature extraction results only restore the RDMB feature extraction results through downsampling to avoid extra computational overhead. Therefore, CAMVA using MARR_S can effectively accelerate CNN accelerators.

D. Dynamic Feature Restoration Strategy

The MARR_S can prune RDMB features in multi-views to enhance input sparse features, accelerating CNN accelerator inference performance. However, the pruning may result in missing certain features, particularly RDMB feature extraction results. To accelerate the CNN accelerator without affecting detection accuracy, two methods can be used: static restoration and dynamic restoration to restore the missing features. Static restoration refers to reusing the $\mathrm{RTMB}^{\mathrm{i}}$ feature extraction results in the specified neural network layer under static configuration to restore all matching $\mathrm{RDMB}^{\mathrm{i}}$ feature extraction results. Dynamic restoration refers to reusing the $\mathrm{RTMB}^{\mathrm{i}}$ feature extraction results under specific conditions, and dynamically restoring each RDMB feature extraction result in each group of matching macroblocks on the optimal neural network layer ($\mathrm{conv_i}$), where i=$\{1,2,\ldots,n\}$ and n is the number of convolutional layers. The layer responsible for the feature restoration is called the feature restoration layer.

In previous research, EVA^2 [36] selected the final convolutional layer of CNN as the optimal feature restoration layer for reconstructing other input features. The static method may give rise to two restoration cases in this paper when restoring missing features. 1) The restoration results do not accurately represent the RDMB feature extraction results due to the delayed restoration. 2) The premature restoration of the RDMB feature extraction results leads to suboptimal performance enhancement. The reason is that the dimensions of RDMBs in each group of matched macroblocks vary, and a uniform static configuration cannot fit all RDBMs. Therefore, the static method is unsuitable for the proposed approach.

In light of this, the paper proposes a dynamic feature restoration strategy that assesses the correlation between the dimensions of RDMBs in the input of each CNN layer and the dimensions of filters in corresponding layers. Based on this correlation, an optimal feature restoration layer is selected for each RDMB in the CNN. The strategy utilizes RTMB feature extraction results to restore matched RDMB feature extraction results in other views, achieving superior performance enhancement with minimal accuracy impact. Specifically, the dynamic feature restoration strategy assesses the correlation between the dimensions of the $\mathrm{RDMB}^{\mathrm{i}}_{\mathrm{v},1}$ ($\mathrm{c_d}$, $\mathrm{w_d}$, $\mathrm{h_d}$) of $\mathrm{view_v}$ at the convolutional layer ($\mathrm{conv_i}$) and the dimension of the filter_{i+1} ($\mathrm{c_k}$, $\mathrm{w_k}$, $\mathrm{h_k}$) at the subsequent convolutional layer (conv_{i+1}). If $\mathrm{w_d} < \mathrm{w_k}$ or $\mathrm{h_d} < \mathrm{h_k}$, it indicates that the $\mathrm{RDMB}^{\mathrm{i}}_{\mathrm{v}}$ feature extraction results will be integrated with its neighboring features, leading to the loss of independence of the $\mathrm{RDMB}^{\mathrm{i}}_{\mathrm{v}}$ feature extraction results in conv_{i+1}. The judgment condition triggers a critical case to restore the $\mathrm{RDMB}^{\mathrm{i}}_{\mathrm{v}}$ feature extraction results. Fig. 5 illustrates the dual-view triggered feature restoration mechanism. $\mathrm{View_a}$ and $\mathrm{view_b}$ contain macroblock pairs $\{\mathrm{RDMB_{a,1}}, \mathrm{RTMB_{a,2}}\}$ and $\{\mathrm{RDMB_{b,1}}, \mathrm{RTMB_{b,2}}\}$ respectively. After processing through the (i-1)-th convolutional layer ($\mathrm{conv_{i-1}}$), both

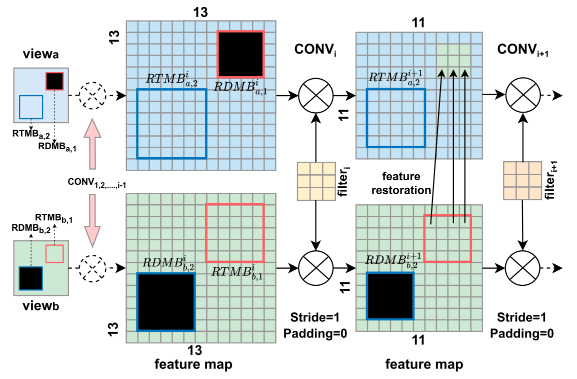


Fig. 5. An example of the dynamic feature restoration in two views.

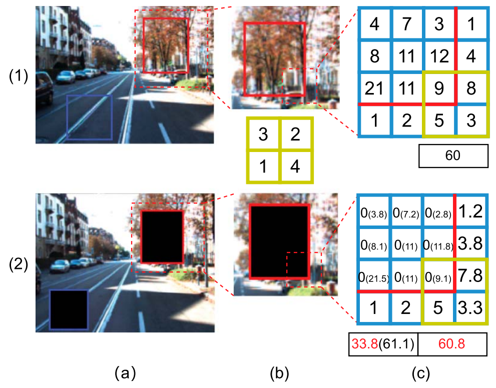


Fig. 6. An example of the edge feature error correction.

views yield output features with dimensions (C,13,13). The matched macroblock pairs are defined as $MB_1=\{RDMB^{i}_{a,1}, RTMB^{i}_{b,1}\}$ and $MB_2=\{RTMB^{i}_{a,2}, RDMB^{i}_{b,2}\}$. Following processing by $conv_i$, the output dimensions for each macroblock are: $RDMB^{i}_{a,1}$ (C,2,2), $RTMB^{i}_{a,2}$ (C,6,6), $RTMB^{i}_{b,1}$ (C,4,4), and $RDMB^{i}_{b,2}$ (C,9,9). The dynamic feature restoration strategy evaluates the dimensional compatibility between macroblocks and filter_{i+1}, determining that only $\mathrm{RDMB}^{i+1}_{a,1}$ triggers feature restoration due to sufficient dimensional matching, while $\mathrm{RDMB}^{i+1}_{b,2}$ fails to trigger due to inadequate dimensional compatibility. Thus, $conv_{i+1}$ is the optimal feature restoration layer for $RDMB_{a,1}$, enabling dynamic feature reconstruction for pruned macroblocks.

The dynamic feature restoration strategy is suitable for the study because MARR_S retains the largest macroblock, RTMB, among a group of matched macroblocks. MARR_S enables the dynamic feature restoration strategy to reuse the RTMB feature extraction results in a down-sampling manner to restore the RDMB feature extraction results, achieving optimal performance enhancement while minimizing accuracy degradation. Each RDMB has its own optimal feature restoration layer, except for cases where the dimension change magnitude fails to trigger the restoration strategy. In such cases, the last convolutional layer serves as the default restoration layer.

E. Error Correction Strategy

The dynamic feature restoration strategy detects changes in the dimensions of RDMBs and reuses the matching RTMB features to restore the RDMB features based on the judgment condition of feature restoration dynamically. This approach restores the missing features while disregarding the errors introduced by edge features, which need to be promptly corrected, otherwise they will have a detrimental impact on target detection accuracy. Edge features denote the outputs obtained after processing input features, including general and RDMB features, through feature extraction. General features encompass those other than RDMB features in the input. To address this issue, the paper proposes an operation for correcting edge feature errors. Specifically, the downsampling of the RTMB feature extraction unaccumulated multiplication results (feature$\times$weight) is then utilized to correct the RDMB edge feature errors based on the correlation between RTMB and RDMB during the inference process.

Fig. 6 illustrates an example of edge feature error correction. Fig. 6(a)–(1) and (2) represent the left and right views, respectively. In the left view, the solid red box represents RTMB, and its partial features are displayed in Fig. 6(c)–(1). In the right view, the solid red box represents RDMB, and its partial features are displayed in Fig. 6(c)–(2). Fig. 6(b) shows magnified key regions (dashed red boxes) and utilizes a 2×2 filter (yellow boxes) to extract features. Fig. 6(c) depicts the edge feature extraction operation in both views. Fig. 6(c)–(1) shows the feature extraction operation on an input that includes general and RTMB features, resulting in a feature extraction result of 60. Fig. 6(c)–(2) illustrates the edge feature extraction operation, which is represented by the intersection area of yellow and red boxes, where the input consists of general and pruned RDMB features, leading to a feature extraction result of 33.8. The data enclosed in parentheses represent the RDMB features prior to pruning in Fig. 6(c)–(2). In the example, the same computation method is used to demonstrate the error caused by pruning. Before pruning, the features extracted from the input of the right view produce a result of 61.1. In this case, feature (9) at coordinates (2,2) in the left view matches with feature (9.1) at coordinates (2,2) in the right view. The error between features before and after pruning is 27.3, indicating a significant error between matched feature extraction results in pruned views. The example reuses the multiplication result (27) of feature (9) in the left view coordinates (2,2) to correct the result of the pruned feature in the right view coordinates (2,2), avoiding adversely affecting subsequent inference. After correction, the feature extraction result is 60.8 with an error rate of only 0.4%. This subsection demonstrates the effective error correction capability of the operation for edge features discussed in Section IV-C regarding its hardware implementation.

IV. Architecture for CAMVA

This section presents the CAMVA workflow in pseudocode (Algorithm 2), with subsequent subsections providing detailed explanations of its functionality. As an extended architecture for accelerating multi-view tasks in CNN accelerators, CAMVA dynamically restores pruned approximate feature extraction results across multiple views, thereby accelerating multi-view inference without compromising accuracy. The implementation

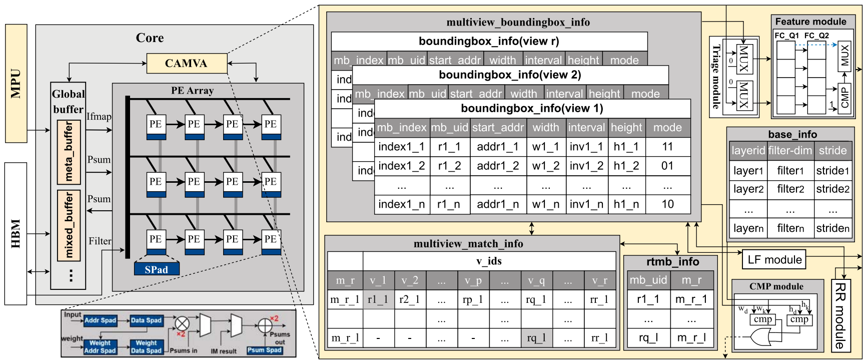


Fig. 7. CAMVA: An extension architecture of the CNN accelerator for multi-view accelerations.

Algorithm 2 CAMVA-CNN Accelerator with Feature Pruning

```
inpput: meta_data(static, dynamic), features
output: Skipping computation and restoring features
 1: Meta_Fetcher loads meta_data during weight loading
 2: CAMVA updates $(mbi, mmi, ri)$ via trigger mechanism
 3: Loading features from HBM by filter size
 4: Classify features: $\{\text{GENERIC}, \text{RDMB}\}$
 5: $FC\_Q1 \leftarrow$ Push feature values
 6: $FC\_Q2 \leftarrow$ Push type codes (1:GENERIC, 0:RDMB)
 7: mixed_buffer $\leftarrow \{FC\_Q1[i] \mid FC\_Q2[i] = 1\}$
 8: PE Array extracts feature maps
 9: if $FC\_Q1.len ==$ count$(FC\_Q2, 1)$ then
10:     No-skip computation
11: else if count$(FC\_Q2, 1) == 0$ then
12:     Complete-skip computation
13:     Reuse Intermediate reuslts and restore features
14: else
15:     Incomplete-skip computation
16:     Reuse Intermediate reuslts and restore features
```

of CAMVA within a CNN accelerator [10] is illustrated in Fig. 7.

A. Metadata

The CAMVA metadata is divided into static and dynamic data. Static data refer to the structured data of each CNN layer, such as filter size and strides, which are written into the static data table (base_info (bi)) during compilation time. Dynamic data refer to the macroblock (MB) data of each view, including the size, properties, and multi-view matching relationship of the MB in each layer, which are generated by meta_product_unit (MPU). Dynamic data tables (multiview_boundingbox_info (mbi), multiview_match_info (mmi), rtmb_info (ri)) are written when weights are updated. Note that the MB size may change after extracting features.

MPU and CAMVA present the producer-consumer strategy. As a metadata producer, MPU efficiently writes dynamic data into the meta_buffer using Direct Memory Access (DMA) technology. As a consumer, CAMVA uses the meta_fetcher to retrieve data from the meta_buffer based on address index and then updates the dynamic data table using the macroblock unique identifier (mb_uid). CAMVA employs a trigger mechanism to update metadata, which is activated when the current layer is computed and weight_fetcher is ready to load weights for the next layer. The metadata update occurs during weight loading without requiring additional clock cycles. The meta_buffer, introduced by CAMVA, is additional storage and a component of the global buffer.

MPU generates metadata in the following steps. Firstly, it loads SIFT features from DRAM using SIFT-specific accelerators (SSAs). Then, it matches the features across multiple views using Algorithm 1 to establish view-feature relationships. Next, it clusters the features of each view using DBSCAN and creates a MB for each cluster. Fourthly, it establishes view-MB relationships based on the view-feature relationships, resulting in mmi formation. Additionally, it creates a boundingbox_info (bbi) for each view by mapping internal and external storage addresses. The tables form mbi in multi-view. Each bbi contains parameters such as mb_index, mb_uid, start_addr, w_t, h_t, interval, and mode representing row index, macroblock unique identification, start address of macroblock features, width, height, feature address interval, and mode respectively. Finally, MARR_S is utilized to identify RTMBs within the group of matched MBs and record their corresponding mb_uids in ri. This allows retrieval of RTMB's mb_uid from the matched MB group. The method is implemented in this paper to evaluate MPU performance on multi-view autopilot datasets at various sizes. The running time (RT) denotes the time cost of MPU to process one frame, and the frame rate (FR) represents the number of images processed by MPU per second. FR is calculated using Eq. (2):

$$FR = \frac{1}{RT} \times n, \tag{2}$$

where n refers to the number of cameras involved in multi-view tasks. The evaluation results are presented in Table II.

CAMVA consumes metadata from the meta_buffer and updates the dynamic data table by loading the MB information of each view into mbi, the matched MB information of each group into mmi, and the mb_uid of RTMB into ri. After computing layer feature extraction, certain data in the dynamic data table, such as MB size, interval, and MB index, may become outdated. Updating the MB index is crucial because RDBMs' indexes

TABLE II
THE FR AND RT OF MPU FOR VARIOUS IMAGE SIZES

Image size	KITTI (binocular cameras)		nuScenes (six cameras)	
	RT (ms)	FR (fps)	RT (ms)	FR (fps)
600×600	14.8	135.14	5.02	1195.22
300×300	5.21	383.88	2.82	2127.66
416×416	9.02	221.73	3.88	1546.39
640×640	14.73	135.78	5.28	1136.36

are invalidated due to feature restoration of RDMBs after layer computation. Therefore, it is necessary to remove outdated MB information from mbi to reduce memory storage and access overhead for dynamic data tables.

B. Dataflow

The CAMVA-equipped CNN accelerator loads features from High Bandwidth Memory (HBM) according to the $filter_i$ size ($c \times w \times h$) in $conv_i$. The CAMVA triage module identifies the input feature type based on the matching relationship between MBs in mmi, the mb_uid of RTMB in ri and the address range of each MB feature in bbi. The input feature types are divided into general features, RTMB features and RDMB features. The general and RDMB features constitute the entirety of the view. The RTMB features are contained in the general features and match with other views RDMB features. CAMVA prunes the input features, i.e. input_fetcher prefetches the general features involved in the computation from HBM into FC_Q1 by feature type and address. FC_Q1 stores the general features and the placeholder “0” ① that replaces the RDMB feature. The CAMVA feature module encodes the feature types in FC_Q1 with binary encoding, where “1” denotes general features, “0” denotes RDMB features, and the results are stored in FC_Q2 ②. The CAMVA-equipped CNN accelerator stores input features, such as the Zero-Free Neuron Array format (ZFNAf) [9], in a suitable format based on the data and indexes in FC_Q1 and FC_Q2 to reduce the computational overhead. The CAMVA feature module writes the general features, represented by “1”, into the mixed_buffer while skipping over the placeholders “0” associated with RDMB features using an index offsetting method ③. The FC_Q1, FC_Q2, mixed_buffer, introduced by CAMVA, are additional storage and a part of the global buffer. Additionally, if the input features contain RTMB features, the extraction results are written to the on-chip buffer. The type of computation processed by the PE array in the CNN accelerator is determined by CAMVA based on the encoding results in FC_Q2 ④. The computations encompass three types, which are illustrated in Fig. 8.

No-skip computation The type indicates that the input features are general and stored in FC_Q1, while their binary codes are stored in FC_Q2. All features from FC_Q1 are written to mixed_buffer. In the no-skip computation, these general features are involved in feature extraction. Fig. 8(a) illustrates the data flow for the no-skip computation.

Incomplete-skip computation The type indicates that the input consists of both general and RDMB features, where RDMB features are represented by “0” placeholders and are stored in FC_Q1. After binary encoding, the encoded general and RDMB features are stored as “1” and “0” in FC_Q2, respectively. Therefore, the CAMVA-equipped CNN accelerator stores the general features in a suitable format (e.g., ZFNAf) in mixed_buffer based on the features in FC_Q1 and the feature types in FC_Q2, thereby skipping zero-valued inputs computation by PE arrays. Fig. 8(b) depicts the data flow for incomplete-skip computation.

Complete-skip computation The type indicates that the input features are the RDMB features (w×h), and the placeholders are used to replace the RDMB features. These placeholders are stored in FC_Q1. After undergoing binary encoding, the binary codes are stored FC_Q2. Consequently, during the complete-skip computation, the RDMB features are not involved in any computations. The CAMVA controls the pointer to skip $w \times h$ placeholders by index, and these placeholders do not occupy any computation cycles in PE arrays. Fig. 8(c) depicts the data flow for the complete-skip computation. Section IV-C describes the hardware architecture in detail regarding the restoration of the RDMB feature extraction results.

Consequently, CAMVA improves the performance of the CNN accelerator by pruning the RDMB features so that the CNN accelerator, which supports the data flow optimization method for one-sided sparse feature maps [9], [37], can skip the computation of approximate redundant features in multi-view without affecting the normal pipeline execution. However, to mitigate the impact of pruning on target detection accuracy, CAMVA dynamically restores the RDMB feature extraction results based on filter and MB sizes using the RTMB feature extraction results.

C. Feature Restoration

Feature restoration consists of three parts: 1) retrieving the matched features, 2) correcting the edge feature errors in real-time, and 3) restoring the RDMB features.

1) Retrieving the matched features: Feature restoration hinges on fast and accurate reuse of RTMB features while maximizing CNN accelerator speedup. CAMVA employs downsampling to load RTMB features matching RDMB features from memory based on MARR_S, with the downsampling interval calculated from the matched MB size. For instance, to restore RDMB features in view$_w$ using RTMB features in view$_v$, assuming their sizes are (C,w$_t$,h$_t$) and (C,w$_d$,h$_d$), respectively, the interval is computed as Eq. (3):

$$interval_w = \frac{w_t}{w_d}, interval_h = \frac{h_t}{h_d}. \quad (3)$$

RTMB features are stored in the row-major order, with each row stored consecutively. Therefore, CAMVA is capable of extracting RTMB features from memory that approximate RDMB features at regular intervals of k elements, where k=$m \times interval_h$+$n \times interval_w$($n \in$[0,w_d-1], $m \in$[0,h_d-1]).

The steps of feature retrieving and matching between MBs in CAMVA are as follows. Firstly, CAMVA retrieves the bbi of view$_v$ from the mbi. Secondly, the mb_uid of the loaded feature’s corresponding MB in view$_v$ can be deduced by referencing the MB feature address range from bbi. Thirdly, using the

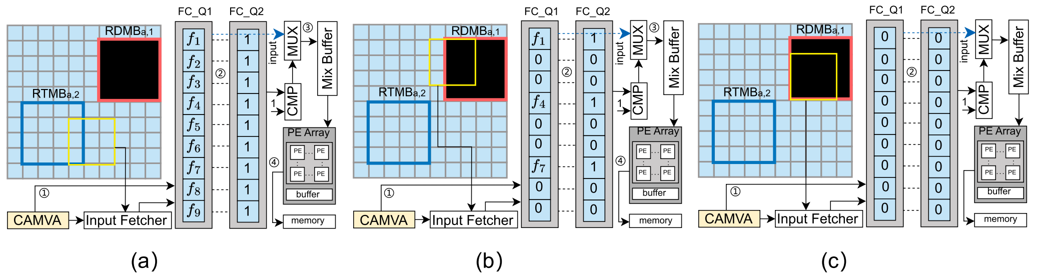


Fig. 8. The three computations types in the CAMVA-equipped CNN accelerator. The yellow box denotes the filter. The red and blue boxes denote macroblocks.

mb_uid of RDMB in view$_v$, CAMVA determines the matched RTMB's mb_uid in view$_w$, based on the mmi. Subsequently, based on the mb_uid of the RTMB in view$_w$, CAMVA determines the address range of RTMB features from bbi. Finally, CAMVA extracts RTMB results that match RDMB results according to the interval.

2) Correcting the edge feature errors: The CAMVA-equipped CNN accelerator loads input features from the HBM and performs the incomplete-skip computation. The input features consist of the general and RDMB features. The output features comprise edge features. In the incomplete-skip computation, the RDMB features are not involved in the feature extraction, which makes the edge features have errors compared to the actual results. However, these errors must be corrected to minimize the pruning operation impact on the detection accuracy. To bridge these errors, two modules are proposed: the loading RTMB features module (LF module) and reusing the extraction results module (RR module) to accommodate diverse cases of edge feature restoration.

LF module In the incomplete-skip computation, RDMB features are involved in feature extraction before the matched RTMB features, which means CAMVA cannot reuse the RTMB feature extraction results to correct errors in the edge features. The reason is the absence of on-chip memory-stored RTMB feature extraction results that approximate the RDMB feature extraction results. To address the issue, CAMVA employs the LF module to load RTMB features from other views that match pruned RDMB features into the computation process. Specifically, the CAMVA-equipped CNN accelerator loads general features into PE arrays from memory and uses the LF module to load matching RTMB features into PE arrays. Then, the RTMB feature extraction results are stored in the on-chip buffer during the feature extraction. The method corrects errors and facilitates an accumulative construction of the RTMB feature extraction results group in the on-chip buffer. The RTMB features can be reused multiple times by the RR module after being computed once.

RR module In the incomplete-skip computation, RTMB features are extracted before the matched RDMB features. General features are also computed. The pruned RDMB features do not require the LF module to load the matching RTMB features for computation but can directly access and reuse the RTMB feature extraction results from the on-chip buffer. The RR module retrieves the feature address range of each RTMB in mbi to determine matching RTMB features with the RDMB features, followed by retrieving addresses for matching RTMB feature extraction results based on the address mapping relationship in the memory controller. The results are accumulated in the accumulator after being extracted by PE arrays; otherwise, the RR module invokes the LF module.

Fig. 9 illustrates a simplified example of error restoration for incomplete-skip computation in two views (view$_a$, view$_b$). Fig. 9(a) shows edge feature extraction in a CAMVA-equipped CNN accelerator with RTMB features (instead of RDMBs), while Fig. 9(b) shows the error between restored and actual edge features via RTMB result reuse. Red boxes in Fig. 9(a)–(1) and 9(b)–(1) denote RTMB$_1$ of view$_a$ and matching RDMB$_1$ of view$_b$, respectively; yellow matrix boxes represent filters. Input features are shown in Fig. 9(a)–(2) and 9(b)–(2), with black boxes indicating computation-involved features. For feature extraction of view$_b$ before view$_a$ in Fig. 9(a)–(3): First, the accelerator loads view$_b$ general features (1.2, 3.8) ① and RTMB$_1$ features (3, 12) matched to view$_a$'s RDMB$_1$ ② into the on-chip buffer, extracting the results of edge features in the PE array ③. Second, the results (9, 2.4, 12, 15.2) are stored in the intermediate result buffer (IM buffer), while the accumulator performs feature accumulation and outputs to the output buffer. The IM buffer operates as an on-chip buffer without occupying additional PE array storage space. When feature reuse is required, CAMVA implicitly retrieves data from the global buffer to the on-chip buffer through the feature loading process. Third, for view$_a$ feature extraction, since (3, 12) are precomputed, CAMVA controls the input fetcher to load the remaining features (1, 4) ④ for PE array computation ⑥, storing results in the IM buffer and triggering RTMB reuse ⑤. Finally, the accumulator outputs the accumulated features. For feature extraction of view$_a$ before view$_b$ in Fig. 9(b)–(3): The CAMVA-equipped CNN accelerator first loads view$_a$ general features (3, 1, 12, 4) ① for PE array extraction ②, storing results (9, 2, 12, 16) in the IM buffer. Next, for view$_b$, CAMVA loads general features (1.2, 3.8) ③, computes (2.4, 15.2) in the PE array ④, and stores them in the IM buffer. Finally, CAMVA reuses results (9, 12) and the accumulator accumulates the results (9, 2.4, 12, 15.2) for output.

3) Restoring the RDMB features: The CAMVA-equipped CNN accelerator loads input features from HBM, where the addresses of RDMB within the input features exhibit continuity and regularity. By exploiting these characteristics, the

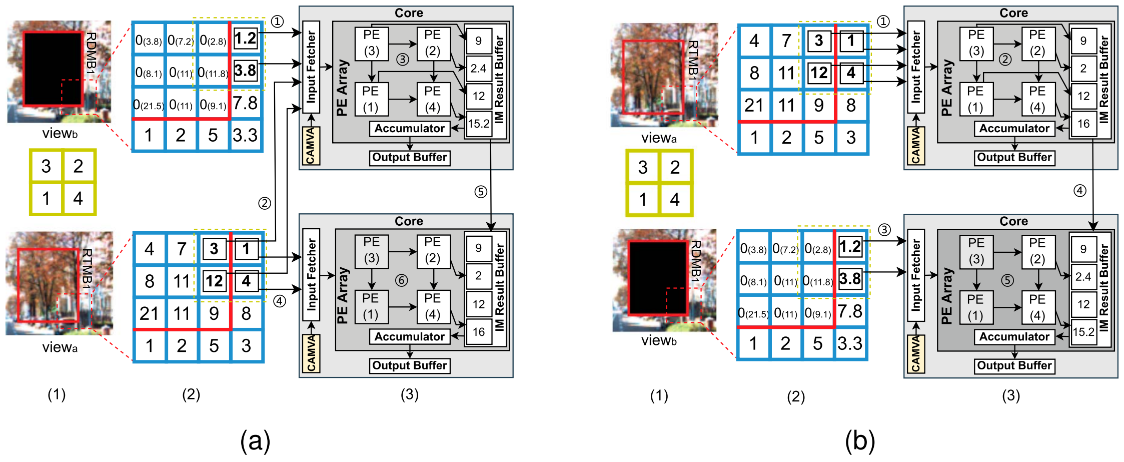


Fig. 9. A simple example of errors restoration. (a) Load RTMB features. (b) Reuse RTMB feature extraction results.

CAMVA-equipped CNN accelerator enables complete-skip computation of RDMB features, reducing computational and memory access overhead associated with inference. However, the complete-skip computation results in missing representations for RDMB feature extraction in the output, potentially compromising detection accuracy. Unlike edge features, which have errors that need to be restored after each feature extraction, RDMB features retain their ability to represent themselves across successive convolutional layers until they lose independence. Feature restoration is necessary before complete loss of independence occurs, a crucial case proposed in Section III-D. The compare (CMP) module of CAMVA assesses the dimension relationship between RDMBs and filters to identify critical cases for each RDMB and determine the corresponding feature restoration layer. Upon being triggered by an RDMB for each critical case, CAMVA utilizes both the LF module and RR module at the feature restoration layer to effectively restore feature extraction results for that specific RDMB, mitigating any impact on detection accuracy caused by pruned features. Restoring RDMB features is a special example of edge feature restoration that allows multiple complete-skip computations before triggering a critical case.

V. Experimental Methodology

The study evaluates the impact of CAMVA operations on the accuracy of four target detection neural networks and CAMVA's performance optimization for CNN accelerators on the multi-view autopilot datasets. The results indicate that CAMVA enhances the performance of CNN accelerators in multi-view tasks and has little impact on accuracy.

A. Validation on Target Detection Accuracy

The subsection outlines the steps to validate the impact of CAMVA feature pruning and dynamic restoration operations on accuracy. The experiment is conducted using an Intel i5-12500H CPU with 16 GB RAM and an NVIDIA Tesla A100 80 GB int the experimental environments. Firstly, SIFT features of the autopilot datasets (KITTI and nuScenes) are extracted by software. Secondly, matched SIFT features are filtered based on the hyperparameter T, determined by the method described in Section III-A, to remove incorrect matches. Higher correct match rates and more matched features are achieved with T_{KITTI}=0.54 and $T_{nuScenes}$=0.597 on the datasets, although a few remaining incorrect matches slightly affect accuracy. Thirdly, DBSCAN clusters SIFT features within each view to construct macroblocks. Finally, CAMVA feature pruning and dynamic restoration operations are implemented during the inference of four target detection neural networks, with accuracies evaluated on the datasets.

The presence of mismatched features during macroblock construction can lead to the matching of dissimilar macroblocks, which degrades the target detection accuracy due to the CAMVA pruning and restoration operations for the macroblocks. To mitigate this issue, the paper utilizes Perceptual Hashing ($pHash$) [38] and Hamming Distance (Ham) [39] to calculate the similarity degree (SD) of macroblocks. It introduces a hyperparameter (HT) as the threshold for filtering out these macroblocks, reducing the impact on accuracy caused by CAMVA operations. SD is calculated by Eq. 4:

$$SD = 1 - \frac{Ham(pHash(RTMB), pHash(RSMB))}{64}. \quad (4)$$

The paper defines the set of HT, denoted as HT_Set={0.1, 0.2,..., 0.9, 1.0}, where $HT_i \in HT\_Set$ and $i \in \{0,1,2,\ldots,9\}$. Only matched macroblocks with $SD \geq HT_i$ are pruned. The KITTI and nuScenes datasets are partitioned into separate training and testing sets for validation accuracy. Four target detection neural networks (FasterRCNN, SSD, Yolo-v3, and Yolo-v5) are trained on the training sets to establish base models. Subsequently, their accuracies are evaluated on the testing sets as baselines. Finally, the impact of CAMVA operations on accuracy is assessed by implementing feature pruning and restoration operations on the testing sets.

To verify the universality and effectiveness of the proposed method, experiments are conducted on three 3D object detection networks (RTMDet-s [40], Mono3D [41], YoloStereo3D

[42]) for the car category under easy, moderate, and hard difficulty levels using the KITTI 3D dataset. The dataset is divided into a training subset and a validation set. Models are trained on the training subset, with final performance evaluated on the validation set.

B. Modeling CAMVA

CAMVA consists of the Feature module, Triage module, CMP module, LF module, and RR module, which implement at a clock frequency of 250 MHz in RTL. The design is synthesized using Synopsys Design Compiler (DC) with the TSMC 130 nm layout and wiring process. Additionally, CAMVA extends the global buffer to store mbi, mmi, ri and bi. CACTI 6.0 tool [43] is used to estimate the extra area occupied by memory.

1) SpeedUp: This study develops a cycle-accurate simulator org_sim based on the Eyriss-v2, with an integrated CAMVA mechanism forming camva_sim for convolutional accelerator evaluation. The experimental dataset is derived from KITTI 2D benchmark, and the dataset initially constructs original subsets org_subsets (subset10, subset20) with negligible sparsity. After pruning, we generate camva_subsets (subset10', subset20') exhibiting 9.6% and 19.8% sparsity respectively. Baseline cycles (cycle_org) are measured via org_sim processing org_subsets, while optimized cycles (cycle_opt) are obtained from camva_sim evaluating pruned subsets. Layer-wise speedup ratios (SU_i) are defined as Eq. 5:

$$SU_i = \frac{cycle\_org_i}{cycle\_opt_i}, i = (1, 2, ..., n), \tag{5}$$

where $cycle\_org_i$ is a baseline clock cycle overhead in $conv_i$, $cycle\_opt_i$ is a optimized clock cycle overhead in $conv_i$ and n is the number convolutional layers. While the neural network speedup (nn_SU) is calculated by Eq. 6:

$$nn\_SU = \frac{\sum_{i=0}^{n} cycle\_org_i}{\sum_{i=0}^{n} cycle\_opt_i}. \tag{6}$$

2) Energy Analyze: The paper analyzes and evaluates the energy consumption associated with three phases, highlighting the energy-saving benefits of CAMVA: SIFT feature extraction, MVDMA execution, and inference. Firstly, we analyze the energy consumption of SIFT feature extraction in the SIFT-specific accelerator (SSA) [32] for various image sizes. The SSA uses a 130 nm 1P6M CMOS process, occupies a chip area of 32 mm^2, and consumes 9.6 mJ/frame when processing 720P images (1280× 720). Secondly, we monitor CPU power consumption during MVDMA execution using HWiNFO64 to evaluate its energy usage. Finally, we employ org_sim and camva_sim to count MAC operations and data accesses for four object detection neural networks in org_subset and camva_subset. Based on the energy analysis framework [27], we evaluate the Energy Conservation Ratio (ECR) of CAMVA-equipped Eyeriss-v2 in each convolutional layer of these networks. We also analyze the energy consumption breakdown for each network. Eyeriss-v2 incorporates a three-level memory hierarchy (DRAM, SRAM, SPad) with access costs of 200×, 6×, and 2× relative to a MAC operation. Using Eyeriss-v2 parameters (200 MHz, 192 PEs, 8-bit, 253.20 GOPS/W for AlexNet, 102.10 fps for AlexNet), we calculate theoretical energy consumption and frame rates for various neural networks. Then, we incorporate CAMVA's energy-saving effects to evaluate system-wide energy overhead. Each convolution layer's energy consumption is calculated as EC$_i$=200×DRAM + 6×SRAM + 2×SPad + 1×MAC. ECR is defined as (EC$_i$-EC'$_i$)/EC$_i$, where EC$_i$ and EC'$_i$ represent normalized energy consumption for Eyeriss-v2 and CAMVA-equipped Eyeriss-v2, respectively.

VI. Experimental Results

The section demonstrates the performance advantages of CAMVA, evaluates its speedup and energy consumption using the simulators (org_sim and camva_sim), and analyzes the power and area of CAMVA through DC.

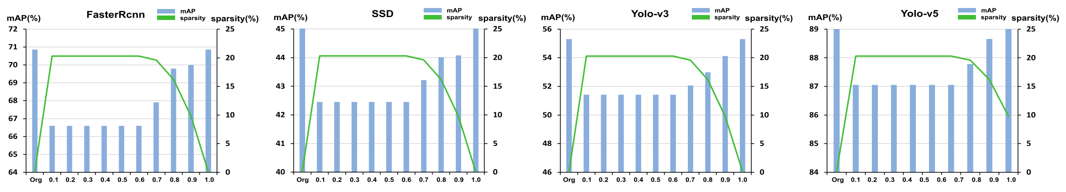
A. Accuracy

The multi-view autopilot datasets (KITTI and nuScenes) are used to simulate two multi-view tasks, with the data pruned using techniques proposed in Section III-B and Section III-C. The baseline accuracies of the four target detection neural networks, along with the accuracies after introducing the CAMVA operations (HT=0.6), are presented in Table III. After introducing the CAMVA operations, there is a 20.3% increase in average sparsity for images from the KITTI dataset and a 10.9% increase for images from the nuScenes dataset, with a corresponding decrease in accuracy by 2.29%∼4.26% and 0.37%∼1.11%, respectively. The difference in accuracy drop between the two tasks is due to the difference in view similarity, which is 88.51% for the KITTI task and 42.66% for the nuScenes task. The experimental results suggest that the impact of CAMVA on target detection accuracy is influenced by the similarity between views in multi-view tasks, as matching SIFT features in highly similar multi-view tasks introduces more errors. In Table III, parameters such as mAP, mAA, mR, and mF1 correspond to detection ability, classification ability, recall rate, and overall recognition capability. The relationship between multi-view sparsity and target detection accuracy is shown in Fig. 10 after introducing the CAMVA operation. The pruning threshold $HT \in HT\_Set$ is a crucial parameter affecting this relationship. The experimental results demonstrate that the sparsity remains constant within the range of HT ∈[0.1,0.2,. . .,0.6]. When $HT \geq$ 0.6, the sparsity gradually decreases until it reaches 0, leading to target detection accuracy initially stabilizing and subsequently converging towards the baseline.

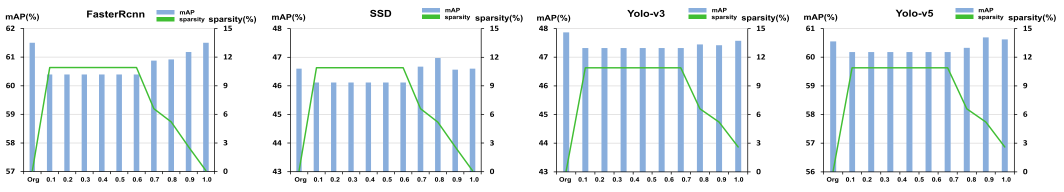
To validate the proposed method, we conducted 3D detection accuracy experiments on the KITTI dataset for cars at three levels: easy, moderate, hard, under Intersection over Union(IoU)=0.5, for the 3D object detection networks [40], [41], [42]. Table IV compares baseline accuracy with results after introducing CAMVA (Column 1: network; Column 2: baseline accuracy; Column 3: accuracy after introducing CAMVA). After pruning and feature restoration, the accuracy changes of each network are as follows: a decrease of 0.4%∼1.97% under easy, 0.22%∼1.1% under moderate, and 0.09%∼–0.22% under

TABLE III
THE COMPARISON OF BASELINES AND CAMVA RESULTS ACROSS DATASETS IN THE DIFFERENT TARGET DETECTION NEURAL NETWORKS

network	Baseline				Introduced the CAMVA operations			
	mAP (%)	mF1	mR (%)	mAA (%)	mAP (%)	mF1	mR (%)	mAA (%)
KITTI								
FasterRCNN	70.86	0.63	77.34	54.10	66.60	0.61	72.55	53.08
SSD	45.08	0.35	24.12	84.91	42.45	0.34	23.03	82.49
Yolo-v3	55.30	0.49	38.20	87.24	51.42	0.47	35.93	85.75
Yolo-v5	89.34	0.89	85.28	93.06	87.05	0.87	83.07	91.96
nuScenes								
FasterRCNN	61.50	0.61	62.03	62.28	60.39	0.59	60.37	60.40
SSD	46.60	0.45	32.43	83.44	46.12	0.44	31.95	82.48
Yolo-v3	47.87	0.47	35.52	89.28	47.32	0.46	35.31	89.67
Yolo-v5	60.55	0.56	43.75	91.91	60.18	0.56	43.73	93.50



(a) KITTI.



(b) nuScenes.

Fig. 10. The trend relationship between accuracy and sparsity.

TABLE IV
COMPARISON OF 3D DETECTION RESULTS FOR CARS

Network	AP_{3D} (IoU=0.5)			**Ours** (IoU=0.5)		
	Easy	Mod.	Hard	Easy	Mod.	Hard
RTMDet-s [40]	38.30	28.65	24.34	37.79	28.43	24.56
Mono3D [41]	60.92	42.18	32.02	58.95	41.08	32.06
YoloStereo3D [42]	65.68	41.25	30.42	65.28	40.57	30.33

hard. DA3D shows smaller accuracy changes due to data augmentation during training. YoloStereo3D also maintains stable performance thanks to binocular data use, multi-scale feature extraction, and fusion. Mono3D experiences slower accuracy decline compared to Table III, which is attributed to its training-time data augmentation.

B. Performance

1) SpeedUp: The paper employs clock-accurate simulators (org_sim and camva_sim) to execute target detection networks on the testing subsets. It then calculates the clock cycle of each layer, i.e., cycle_org$_\mathrm{i}$, cycle_opt$_\mathrm{i}$, as well as the system clock cycle of the target detection network, i.e., $sys\_cycle\_org = \sum_{i=1}^{n}(cycle\_org_i)$, $sys\_cycle\_opt = \sum_{i=1}^{n}(cycle\_opt_i)$. The speedup of each layer in CNNs with the CAMVA operations is depicted in Fig. 11, as evaluated according to Eq. 5 and Eq. 6. Further analysis reveals that on subset10, CNNs achieve a nn_SU of FasterRCNN: 1.05×, SSD: 1.08×, Yolo-v3: 1.04×, and Yolo-v5: 1.07×, while on subset20 CNNs achieve a nn_sp of FasterRCNN: 1.13×, SSD: 1.22×, Yolo-v3: 1.34×, and Yolo-v5: 1.23× respectively. The experimental results show that increasing the input sparsity improves the performance of CNNs and their convolutional layers. Moreover, the observed trend in the speedup of layers from the experiments is also related to the CNN architecture.

2) Energy Analyze: The paper uses simulators (org_sim and camva_sim) along with the energy framework [27] to analyze the computational and storage access overhead of CNNs on testing subsets, aiming to evaluate the energy consumption advantages provided by CAMVA. Fig. 11 shows ECR of CAMVA-equipped Eyeriss-v2 for each convolutional layer, which changes in the same trend as SU. Fig. 12 shows the energy breakdown of simulators performing FasterRCNN, SSD, Yolo-v3, and Yolo-v5 on org_subset and camva_subset. Fig. 12(a) illustrates the energy breakdown on subset10 and subset10', while Fig. 12(b) presents it on subset20 and subset20'. The energy breakdown (light) of the CNN on org_subset by org_sim serves as a baseline for evaluation, representing computational and storage access without optimization. The

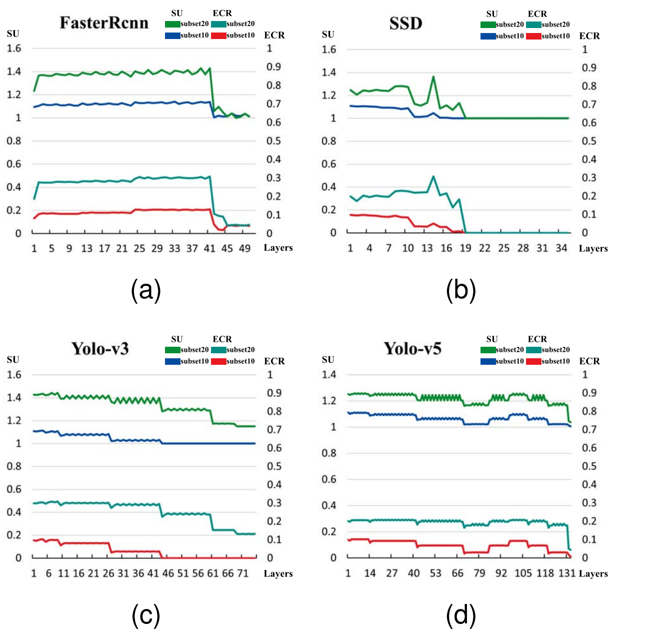


Fig. 11. The SU and ECR of CAMVA-equipped eyeriss-v2 at each neural network CNN layer in the subset10 and subset20.

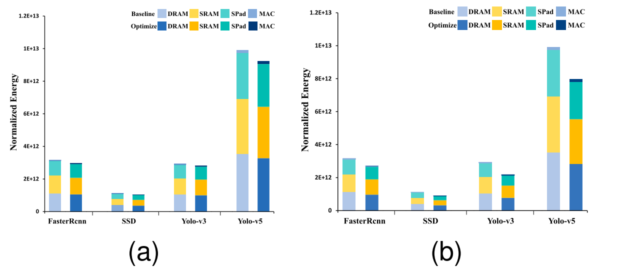


Fig. 12. The energy breakdown in eyeriss-v2 and CAMVA-equipped eyeriss-v2 for the CNNs in subset10 and subset20.

energy breakdown (dark) of the CNN on camva_subset by camva_sim represents optimized results for evaluation, incorporating feature pruning and restoration operations. The overall energy consumption reduction achieved by camva_sim compared to org_sim in each of the four target detection networks is 6.23%, 8.01%, 4.23%, and 7.12% on subset10, and on subset20, the reductions are 14.13%, 19.95%, 25.97%, and 19.89% respectively.

C. End to End Performance

The lowest-performing module in the autopilot system is the performance bottleneck of the entire system. The paper evaluates the performance of SIFT-specific accelerator (SSA), meta_process_unit (MPU), and CNN accelerator (CNNA) according to the data flow execution order. Table V presents the frame rate (FR) for each phase with different image sizes. For 720P images, SSA [32] achieves a FR of 30fps when extracting SIFT features. The second column represents the FR of SSA for other image sizes, using the image proportional scaling method.

TABLE V
THE OVERALL FRAME RATE EVALUATION ON VARIOUS IMAGE SIZES

Image Size	SSA FR(fps)	MPU FR(fps)	CNNA FR(fps)	**Opt-CNNA FR(fps)**
600×600	76.80	135.14	2.80	**3.16**
300×300	307.2	383.88	8.15	**9.94**
416×416	159.76	221.73	6.07	**8.13**
640×640	67.50	135.78	1.51	**1.86**

TABLE VI
THE OVERALL ENERGY COMSUMPTION IMPACT OF CAMVA FOR CNNS

CNN Model	OP [GOP]	SSA	MPU	CAMVA	Inferred EC	**CAMVA-equipped Eyeriss-v2**
		EC	EC	EC	Inferred EC	**Optimiezed EC**
FasterRCNN (600*600)	48.42	+3.75	+1.72	-27.05	+191.43	**164.38**
SSD (300*300)	16.66	+0.94	+0.47	-13.13	+65.77	**52.64**
Yolo-v3 (416*416)	22.38	+1.80	+0.94	-22.95	+88.30	**65.35**
Yolo-v5 (640*640)	90.13	+4.27	+4.21	-70.44	+354.97	**284.53**

The third column represents the FR of the MPU, which is the reciprocal of the RT in Table II. The fourth and fifth columns represent the FR of CNNA and CAMVA-equipped Eyeriss-v2 (Opt-CNNA), respectively. The performance (frame rate (FR) and energy consumption (EC)) of different neural networks on Eyeriss-v2 and CAMVA-equipped Eyeriss-v2 can be inferred according to YC Lu et al.'s approach [44]. The experimental evaluation shows that $FR_{MPU} > FR_{SSA} > FR_{CNNA}$, and CNNA is the performance bottleneck of the system. By integrating CAMVA with CNNA, Opt-CNNA achieves higher performance than CNNA alone.

The paper evaluates energy consumption on subset20. Table VI shows the energy consumption by SSA for feature extraction and MPU for MVDMA on CPU (positive, "+") across models with different input image sizes, as well as the reduced energy consumption of Eyeriss-v2 with CAMVA (negative, "-"). Results show that the energy saved by CAMVA exceeds the additional energy cost of SSA and MPU, demonstrating its effectiveness in reducing energy consumption for CNN accelerators in SSA-equipped autopilot systems.

D. Synthesize

CAMVA is evaluated by DC in the TSMC 130 nm process, consuming 126.72 mW of power and having an area of 0.477517 mm^2. It requires a global buffer to store feature extraction results, mapping relationships between dynamic data tables, and address index entires of features. To meet the storage requirement for simulating multi-view tasks in KITTI and nuScenes, CAMVA is equipped with 32 KB on-chip storage. Since Eyeriss-v2 is implemented in the TSMC 65nm, the paper employs the scaling compensation process described in [36], which involves normalizing the area and power for Eyeriss-v2 to the technology scaling factor. The area for Eyeriss-v2 is 31.86 mm^2 in the TSMC 65 nm process; compensating for the process difference, Eyeriss-v2 occupies approximately 63.72

mm^2 in the TSMC 130 nm process. CAMVA accounts for only 0.75% of the total area, with its on-chip buffer making up about 99.68% of that space usage within CAMVA itself. Furthermore, CAMVA primarily involves basic logic operations while restoring RDMB edge features and RDMB features, resulting in minimal power consumption.

VII. Conclusion

In this study, we propose CAMVA, a method that uses approximate regions in multi-view inputs to enhance the performance and energy efficiency of CNN accelerators without incurring high chip design and manufacturing costs. Experiments show that CAMVA accelerates CNN inference, reduces energy consumption in multi-view scenarios, and maintains accuracy. By leveraging both simultaneous and sequential inputs, CAMVA exploits inter-view redundancy, potentially improving accuracy, a key focus of future work.

References

[1] "Artificial intelligence & autopilot." Tesla. [Online]. Available: https://www.tesla.com/AIAccessed4Nov.2025.
[2] "Driving the autonomous evolution." Mobileye. [Online]. Available: https://www.mobileye.com/.Accessed4Nov.2025.
[3] "Self-driving passenger vehicles." Foresight. [Online]. Available: https://www.foresightauto.com/industries/passenger-vehicles.Accessed4Nov. 2025.
[4] X. Zheng, Y. Li, D. Duan, L. Yang, C. Chen, and X. Cheng, "Multivehicle multisensor occupancy grid maps (MVMS-OGM) for autonomous driving," *IEEE Internet Things J.*, vol. 9, no. 22, pp. 22944–22957, Nov. 2022.
[5] C. Badue et al., "Self-driving cars: A survey," *Expert Syst. Appl.*, vol. 165, Mar. 2021, Art. no. 113816.
[6] G. Babu Naik et al., "Convolutional neural network based on self-driving autonomous vehicle (CNN)," in *Proc. Innov. Data Commun. Technol. Appl. (ICIDCA)*, New York, NY, USA: Springer-Verlag, 2022, pp. 929–943.
[7] Z. Zhou, Q. J. Wu, S. Wan, and X. Sun, "Integrating sift and CNN feature matching for partial-duplicate image detection," *IEEE Trans. Emerg. Topics Comput. Intell.*, vol. 4, no. 5, pp. 593–604, Oct. 2020.
[8] C. Liu, J. Yuen, and A. Torralba, "Sift flow: Dense correspondence across scenes and its applications," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 33, no. 5, pp. 978–994, May 2011.
[9] J. Albericio et al., "Cnvlutin: Ineffectual-neuron-free deep neural network computing," *SIGARCH Comput. Archit. News*, vol. 44, no. 3, pp. 1–13, 2016.
[10] Y.-H. Chen, T.-J. Yang, J. Emer and V. Sze, "Eyeriss v2: A flexible accelerator for emerging deep neural networks on mobile devices," *IEEE J. Emerg. Sel. Topics Circuits Syst.*, vol. 9, no. 2, pp. 292–308, Jun. 2019.
[11] A. Parashar et al., "SCNN: An accelerator for compressed-sparse convolutional neural networks," *SIGARCH Comput. Archit. News*, vol. 45, no. 2, pp. 27–40, 2017.
[12] A. Geiger et al., "Vision meets robotics: The KITTI dataset," *Int. J. Robot. Res.*, vol. 32, no. 11, pp. 1231–1237, 2013.
[13] D. G. Lowe, "Object recognition from local scale-invariant features," in *Proc. 7th IEEE Int. Conf. Comput. Vis.*, 1999, vol. 2, pp. 1150–1157.
[14] X. Chen, H. Ma, J. Wan, B. Li, and T. Xia, "Multi-view 3d object detection network for autonomous driving," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2017, pp. 1907–1915.
[15] R. Szeliski et al., "Image alignment and stitching: A tutorial," *Found. Trends Comput. Graph. Vision*, vol. 2, no. 1, pp. 1–104, 2007.
[16] D. Scharstein and R. Szeliski, "A taxonomy and evaluation of dense two-frame stereo correspondence algorithms," *Int. J. Comput. Vision*, vol. 47, no. 1, pp. 7–42, 2002.
[17] P. Kovesi et al., "Image features from phase congruency," *Videre, J. Comput. Vision Res.*, vol. 1, no. 3, pp. 1–26, 1999.
[18] P. Milanfar, "A tour of modern image filtering: New insights and methods, both practical and theoretical," *IEEE Signal Process. Mag.*, vol. 30, no. 1, pp. 106–128, Jan. 2013.
[19] D. Scharstein, "Matching images by comparing their gradient fields," in *Proc. 12th Int. Conf. Pattern Recognit.,* Piscataway, NJ, USA: IEEE Press, 1994, vol. 1, pp. 572–575.
[20] S. Belongie, J. Malik, and J. Puzicha, "Shape matching and object recognition using shape contexts," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 24, no. 4, pp. 509–522, Apr. 2002.
[21] N. Dalal and B. Triggs, "Histograms of oriented gradients for human detection," in *Proc. IEEE Conf. Comput. Vision Pattern Recognit.*, Piscataway, NJ, USA: IEEE Press, 2005, vol. 1, pp. 886–893.
[22] C. Harris et al., "A combined corner and edge detector," in *Proc. Alvey Vision Conf.*, 1988, vol. 15, pp. 10–5244.
[23] P. Ouyang, S. Yin, L. Liu, Y. Zhang, W. Zhao, and S. Wei, "A fast and power-efficient hardware architecture for visual feature detection in affine-sift," *IEEE Trans. Circuits Syst. I, Reg. Papers*, vol. 65, no. 10, pp. 3362–3375, Oct. 2018.
[24] Z. Fan et al., "ASP-SIFT: Using analog signal processing architecture to accelerate keypoint detection of SIFT algorithm," *IEEE Trans. Very Large Scale Integr. (VLSI) Syst.*, vol. 28, no. 1, pp. 198–211, Jan. 2020.
[25] G. Xiao, K. Li, Y. Chen, W. He, A. Y. Zomaya, and T. Li, "CASpMV: A customized and accelerative SpMV framework for the Sunway TaihuLight," *IEEE Trans. Parallel Distrib. Syst.*, vol. 32, no. 1, pp. 131–146, Jan. 2021.
[26] S. Jung et al., "Learning to quantize deep networks by optimizing quantization intervals with task loss," in *Proc. IEEE Conf. Comput. Vision Pattern Recognit.*, 2019, pp. 4350–4359.
[27] Y.-H. Chen, T. Krishna, J. S. Emer and V. Sze, "Eyeriss: An energy-efficient reconfigurable accelerator for deep convolutional neural networks," *IEEE J. Solid-State Circuit*, vol. 52, no. 1, pp. 127–138, Jan. 2017.
[28] T.-T. Do et al., "Understanding the security and robustness of sift," in *Proc. 18th ACM Int. Conf. Multimedia*, 2010, pp. 1195–1198.
[29] L. Wang and Y. Zhang et al., "On the Euclidean distance of images," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 27, no. 8, pp. 1334–1339, Aug. 2005.
[30] K. G. Derpanis, "Overview of the RANSAC algorithm," *Image Rochester NY*, vol. 4, no. 1, pp. 2–3, 2010.
[31] B. Liu et al., "An energy-efficient SIFT based feature extraction accelerator for high frame-rate video applications," *IEEE Trans. Circuits Syst. I, Reg. Papers*, vol. 69, no. 12, pp. 4930–4943, Dec. 2022.
[32] J. Oh et al., "A 320 mw 342 GOPS real-time dynamic object recognition processor for HD 720p video streams," *IEEE J. Solid-State Circuit*, vol. 48, no. 1, pp. 33–45, Jan. 2013.
[33] A. Ram et al., "A density based algorithm for discovering density varied clusters in large spatial databases," *Int. J. Comput. Appl.*, vol. 3, no. 6, pp. 1–4, 2010.
[34] M. Ester et al., "A density-based algorithm for discovering clusters in large spatial databases with noise," in *Proc. Kdd*, 1996, vol. 96, pp. 226–231.
[35] N. Rahmah and I. S. Sitanggang, "Determination of optimal epsilon (Eps) value on DBSCAN algorithm to clustering data on peatland hotspots in Sumatra," in *Proc. IOP Conf. Ser., Earth Environ. Sci.*, 2016, vol. 31, Art. no. 012012.
[36] M. Buckler et al., "Eva^2: Exploiting temporal redundancy in live computer vision," in *Proc. 45th ACM/IEEE Annu. Int. Symp. Comput. Archit.*, 2018, pp. 533–546.
[37] S. Zhang et al., "Cambricon-X: An accelerator for sparse neural networks," in *Proc. 49th Annu IEEE/ACM Int. Symp. Microarchit.,* 2016, pp. 1–12.
[38] L. Struppek et al., "Learning to break deep perceptual hashing: The use case NeuralHash," in *Proc. ACM Conf. Fairness, Accountability, Transparency*, 2022, pp. 58–69.
[39] M. Norouzi, D. J. Fleet, and R. R. Salakhutdinov, "Hamming distance metric learning," in *Proc. Adv. Neural Inf. Process. Syst.*, 2012, vol. 25, pp.1061-1069.
[40] Y. Jia, J. Wang, H. Pan, and W. Sun, "Enhancing monocular 3-D object detection through data augmentation strategies," *IEEE Trans. Instrum. Meas.*, vol. 73, pp. 1–11, 2024.
[41] Y. Liu, Y. Yixuan, and M. Liu, "Ground-aware monocular 3D object detection for autonomous driving," *IEEE Robot. Autom. Lett.*, vol. 6, no. 2, pp. 919–926, Apr. 2021.
[42] Y. Liu, L. Wang, and M. Liu, "YOLOStereo3D: A step back to 2D for efficient Stereo 3D detection," in *Proc. IEEE Int. Conf. Robot. Autom. (ICRA)*, Piscataway, NJ, USA: IEEE Press, 2021, pp. 13018–13024.
[43] M. Naveen, S. Ali, and S. Vaishnav. "CACTI 6.0." GitHub. [Online]. Available: https://github.com/HewlettPackard/cacti.Accessed4Nov.2025.

[44] Y.-C. Lu et al., "An 176.3 GOPs object detection CNN accelerator emulated in a 28nm CMOS technology," in *Proc. IEEE 3rd Int. Conf. Artif. Intell. Circuits Syst.*, 2021, pp. 1–4.

Junjie Chen (Student Member, IEEE) received the B.S. degree in mechatronics engineering from Hunan University of Arts and Science, China, in 2009, and the M.S. degree in computer technology from Hunan University, Hunan, China, in 2016. He is currently working toward the Ph.D. degree with Hunan University, China. From 2016 to 2020, he worked with CRRC Zhuzhou Institute Company, Ltd and participated in the research and development of industrial big data. His research interests include computer architecture and high-performance computing.

Yan Ding (Member, IEEE) received the Ph.D. degree in computer science from Hunan University, China, in 2021. He is currently an Assistant Professor with Hunan University. His research interests include parallel computing, mobile edge computing, big data, artificial intelligence, and architecture. He has published eight papers in journals and conferences, including Design Automation Conference, IEEE TRANSACTIONS ON PARALLEL AND DISTRIBUTED SYSTEMS, IEEE TRANSACTIONS ON SERVICES COMPUTING, IEEE TRANSACTIONS ON INDUSTRIAL INFORMATICS, *Journal of Parallel and Distributed Computing*, Computers & Security, and 17th IEEE ISPA. He received the Outstanding Paper Award in the 17th IEEE ISPA.

Chubo Liu (Member, IEEE) received the B.S. and Ph.D. degrees in computer science and technology from Hunan University, China, in 2011 and 2016, respectively. He is currently a Full Professor of computer science and technology with Hunan University. His research interests include parallel and distributed computing, computer architecture, artificial intelligence, game theory, and approximation and randomized algorithms. He has published over 40 papers in journals and conferences such as the IEEE TRANSACTIONS ON PARALLEL AND DISTRIBUTED SYSTEMS, IEEE TRANSACTIONS ON CLOUD COMPUTING, IEEE TRANSACTIONS ON MOBILE COMPUTING, IEEE TRANSACTIONS ON INDUSTRIAL INFORMATICS, *IEEE Internet of Things*, ACM ToMPECS, TCS, ISCA, DAC, and NPC. He won the IEEE TCSC Early Career Researcher Award in 2019.

Keqin Li (Fellow, IEEE) received the B.S. degree in computer science from Tsinghua University, in 1985, and the Ph.D. degree in computer science from the University of Houston, in 1990. He is a SUNY Distinguished Professor with the State University of New York and a National Distinguished Professor with Hunan University (China). He has authored or co-authored more than 1200 journal articles, book chapters, and refereed conference papers. He holds nearly 80 patents announced or authorized by the Chinese National Intellectual Property Administration. He is among the world's top few most influential scientists in parallel and distributed computing, regarding single-year impact (ranked
2) and career-long impact (ranked
3) based on a composite indicator of the Scopus citation database. He is listed in Scilit Top Cited Scholars (2023–2025) and is among the top 0.02% out of over 20 million scholars worldwide based on top-cited publications in the last ten years. He is listed in ScholarGPS Highly Ranked Scholars (2022–2024) and is among the top 0.002% out of over 30 million scholars worldwide based on a composite score of three ranking metrics for research productivity, impact, and quality in the recent five years. He received the IEEE TCCLD Research Impact Award from the IEEE CS Technical Committee on Cloud Computing in 2022 and the IEEE TCSVC Research Innovation Award from the IEEE CS Technical Community on Services Computing in 2023. He won the IEEE Region 1 Technological Innovation Award (Academic) in 2023. He was a recipient of the 2022–2023 International Science and Technology Cooperation Award and the 2023 Xiaoxiang Friendship Award of Hunan Province, China. He is a member of SUNY Distinguished Academy. He is an AAAS Fellow, an AAIA Fellow, an ACIS Fellow, and an AIIA Fellow. He is a member of the European Academy of Sciences and Arts. He is a member of Academia Europaea (Academician of the Academy of Europe).