RT-BEV: Enhancing Real-Time BEV Perception for Autonomous Vehicles

Liangkai Liu*, Jinkyu Lee†, and Kang G. Shin*
*Department of Computer Science and Engineering, University of Michigan, USA
†Department of Computer Science and Engineering, Sungkyunkwan University, Republic of Korea

https://github.com/Torreskai0722/RT-BEV

2024 IEEE Real-Time Systems Symposium (RTSS) | 979-8-3315-4026-5/24/$31.00 ©2024 IEEE | DOI: 10.1109/RTSS62706.2024.00031

***Abstract*—Vision-centric Bird's Eye View (BEV) perception has become popular for enhancing the situational awareness of autonomous vehicles (AVs). It uses multiple cameras to create a 360° view, capturing essential details for the vehicle's navigation and decision-making. However, reducing the end-to-end (e2e) BEV perception latency without sacrificing accuracy is challenging due to the lack of co-optimization of message communication and object detection. Prior work either compresses the dense detection model to reduce computation which can hurt accuracy and assume images are well synchronized, or focuses on worst-case communication delay without considering the characteristics of object detection.**

To meet this challenge, we propose `RT-BEV`, the first framework designed to co-optimize message communication and object detection to improve real-time e2e BEV perception without sacrificing accuracy. The main insight of `RT-BEV` lies in generating traffic environment- and context-aware Regions of Interest (ROIs) for AV safety, combined with ROI-aware message communication. `RT-BEV` features an ROI-aware Camera Synchronizer that adaptively determines message groups and allowable delays based on ROIs' coverage. We also develop a ROIs Generator to model context-aware ROIs and a Feature Split & Merge component to handle variable-sized ROIs effectively. Furthermore, a Time Predictor forecasts timelines for processing ROIs, and a Coordinator jointly optimizes latency and accuracy for the entire e2e pipeline. We have implemented `RT-BEV` in a ROS-based BEV perception pipeline and evaluated it with the nuScenes dataset. `RT-BEV` is shown to significantly enhances real-time BEV perception, reducing average e2e latency by 1.5×, maintaining high mean Average Precision (mAP), doubling the number of processed frames, and improving the frame efficiency score (FES) by 2.9× compared to the existing approaches. Moreover, `RT-BEV` is shown to reduce the worst-case e2e latency by 19.3×.

***Index Terms*—BEV perception, region of interests (ROIs)**

I. Introduction

The popularity of vision-centric Bird's Eye View (BEV) perception has surged, enhancing the situation-awareness of autonomous vehicles (AVs) [1]–[3]. BEV perception uses multiple cameras to generate a comprehensive 360° view, capturing crucial details for the vehicle's navigation and decision-making [4], [5], significantly improving AV navigation and decision-making processes [6]. However, achieving real-time BEV perception is challenging due to the high computational demands of processing high-resolution images from multiple cameras [7]–[9]. Real-time processing is crucial for AV safety, as delays can raise safety risks [10], [11]. Innovative solutions are needed to balance real-time performance and accuracy in BEV perception [12].

State-of-the-art (SOTA) approaches often use model compression to trade off some accuracy for speed [13], [14]. Methods like SparseBEV employs sparsity to compress the dense model to be sparse [15], [16], while FastBEV and SparseViT optimize input resolution and network structures for faster processing [13], [17]. However, these methods often suffer from degraded accuracy [18], [19] and overlook the communication delays associated with multi-camera synchronization [20], [21]. Assuming pre-synchronized images is impractical, as each camera's trigger time and communication delays are significant [20]. Other works recognize the importance of message communications, investigating worst-case communication delays [22], [23], but fail to guarantee end-to-end (e2e) BEV perception latency by ignoring the characteristics of object detection.

Considering these issues, we pose a critical question: *Can we co-design message communication and object detection in BEV perception to improve real-time e2e performance without compromising accuracy?* We propose focusing on Regions of Interest (ROIs) processing and enhancing system coordination to optimize the use of computational and communication resources dynamically. Our key insight lies in context-aware generation and processing of ROIs, coupled with ROI-aware message communication of multi-camera inputs. By dynamically adjusting ROIs based on environmental contexts, we can prioritize critical areas and ignore less relevant data, improving efficiency for the e2e pipeline.

Several technical challenges must be addressed. First, the multi-camera system introduces significant delays in synchronizing captured images for BEV perception, potentially causing higher communication delays. Second, modeling ROIs for complex traffic environments must maintain detection accuracy. Third, ROIs in different traffic environments and contexts have variable shapes across cameras, which could incur extra overheads on resource-limited GPU platforms. Lastly, making the trade-off between latency and accuracy is essential to guarantee AV safety.

`RT-BEV` addresses these technical challenges by designing and using several innovative components. The ROI-aware Camera Synchronizer reduces synchronization delays by adjusting queues and allowable delays based on ROIs. The ROIs

979-8-3315-4026-5/24/$31.00 ©2024 IEEE
DOI 10.1109/RTSS62706.2024.00031

Generator is designed to generate ROIs based on driving context and previous detection results from BEV perception. The adaptive Feature Split & Merge module enhances processing efficiency by focusing on critical ROIs and using the previous frame's feature maps for less important areas. The Time Predictor calculates time-to-collision (TTC) and predicts inference times for varying ROI sizes, ensuring timely system adjustments. The central Coordinator manages keyframe frequency and synchronization strategies, balancing latency and detection accuracy for e2e BEV perception.

Our comprehensive testing and evaluation of RT-BEV using a GPU desktop setup and the nuScenes dataset [20] demonstrated its effectiveness. RT-BEV significantly reduces both communication and detection delays, achieving an average e2e latency of 377.6ms, a 1.5× speedup over the 580.6ms in base settings. Additionally, RT-BEV reduces the worst-case e2e latency by 19.3×. For detection accuracy, RT-BEV nearly doubles the number of processed frames of traditional methods, maintaining high mean Average Precision (mAP) and Average Multi-Object Tracking Precision (AMOTP) across various object classes. Furthermore, the frame efficiency score (FES) of RT-BEV shows a 2.9× improvement over SOTA approaches. To the best of our knowledge, RT-BEV is the first framework for BEV perception that significantly enhances e2e real-time performance without sacrificing accuracy.

Overall, this paper makes four main contributions:

- A novel approach to the generation and processing of ROIs in BEV perception. This approach dynamically adjusts ROIs based on environmental and driving contexts, which significantly enhances processing efficiency by focusing on the most critical areas for the AV's navigation and decision-making.
- A flexible synchronization mechanism tailored to dynamic ROIs. This design enables the synchronization of multi-camera images to be flexible and adaptive to dynamically changing ROIs.
- An adaptive BEV perception framework that co-designs communication and computation to balance e2e latency with detection accuracy. The adaptive Feature Split & Merge module, combined with the Time Predictor and Coordinator, focuses computational and communication resources on dynamically determined ROIs.
- Comprehensive evaluation, demonstrating the effectiveness of RT-BEV in real-world scenarios. Implemented in a ROS-based system and evaluated using the nuScenes dataset, our results significantly reduce e2e latency (1.5×) while maintaining a high mAP and AMOTP. Moreover, it achieves a substantial improvement for worst-case e2e latency (19.3×) and frame efficiency score (FES) (2.9×) over SOTA approaches.

II. Background, Motivation, and Problem Statement

A. BEV Perception

Multi-camera BEV perception is vital for AV due to its 360-degree overhead view, enhancing navigation and situational awareness [1], [2], [4], [5], [14], [15]. By integrating multiple cameras, BEV perception provides clear visibility of surroundings, including hidden lanes, obstacles, and pedestrians, thus improving AVs' awareness and decision-making [1].

Figure 1 presents a detailed e2e pipeline for BEV perception utilizing multi-camera systems [6], [12]. The process initiates with the autonomous vehicle's cameras capturing panoramic images [20], [21]. These images are collected in a queue for each camera and synchronized based on their timestamps [24], [25]. This initial phase of capturing and synchronizing images is termed the communication delay. Following synchronization, the images are processed through an image feature extraction stage, incorporating a backbone and neck architecture to produce image feature maps [6], [26], [27]. These maps are then inputted into a BEV encoder, which translates them into a BEV perspective and constructs BEV feature maps [4], [13], [14]. The last stage involves a detection head that utilizes these BEV feature maps to detect and track objects within the environment. The time taken from image feature extraction to object detection by the detection head constitutes the detection delay. The sum of the communication and detection delays represents the total e2e latency for BEV perception.

B. Motivation

Time Constraints. The BEV perception system processes extensive visual data to create an overhead view, which is time-consuming [13], [15], [16]. This challenge is critical in time-sensitive applications where e2e latency can impact safety.

Variable Significance of Raw Data. Not all sensor data in BEV perception are equally important [15], [17]. Dynamic objects like vehicles and pedestrians are more critical than static objects. Focusing on regions of interest (ROIs) can enhance real-time performance, but SOTA systems fail to differentiate data significance, leading to inefficiencies and unnecessary consumption of computational resources.

C. Problem Statement

The challenge in BEV perception for AVs is optimizing real-time performance and maintaining prediction accuracy on resource-limited embedded platforms. SOTA methods consume excessive time and memory by blindly processing all raw images, regardless of relevance, and ignoring the online synchronization of high-frame-rate camera inputs. *This paper aims to develop a technique for efficient ROI processing through communication and computation co-design to enhance real-time capabilities and ensure accurate perception.*

III. Empirical Studies

Through detailed analysis using real-world BEV perception models and datasets, we have made two key observations essential for achieving real-time BEV perception. First, the ROIs that are crucial for autonomous vehicle safety are determined by the driving context and traffic environment, while ROIs' variable sizes also bring challenges for real-time processing.

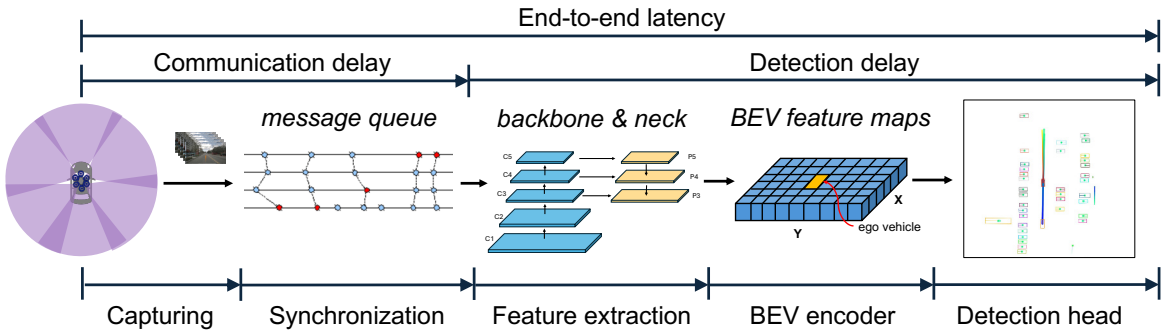

Fig. 1. End-to-end pipeline for BEV perception.

Second, effective multi-camera BEV perception requires time-synchronized images, but the associated capturing and synchronization delays are non-negligible.

A. Environment-Aware ROIs

SOTA BEV perception models tend to process sensory input uniformly throughout the visual field, which can be computationally expensive and inefficient [4], [5], [28], [29]. In reality, not every pixel is equally important for AV's safety [17], [30]. From comprehensive experiments, we found the ROIs that are essential to AV's safety are environment- and context-aware.

In urban environments with complex intersections and heavy pedestrian traffic, the focus should be on crosswalks and sidewalks, while on highways, attention may shift toward the front and nearby vehicles. Additionally, during forward driving, attention is primarily directed toward the front, whereas side views become crucial during lane changes, and rear views are considered when driving in reverse. This observation of regional importance aligns with typical human driving behaviors. Figure 2 illustrates an example region of interest (ROI) using the UniAD model with the nuScenes dataset [6], [20]. The left side of the figure displays the BEV perception results as the ego vehicle moves forward through a parking lot, with another vehicle driving behind. The right side of the figure shows the ROI in each camera to cover all objects, which is less than half of the original size. Considering the vehicle is moving forward and all surrounding vehicles are static, the actual ROI can be reduced to half of the front camera's view (1/12 of the original size). This example clearly illustrates how ROIs can be dynamically adjusted based on traffic environment and driving context.

Timing Benefits with ROIs. The environment-aware ROI ensures that the perception system conserves computational resources by avoiding irrelevant or static areas that do not influence the vehicle's operational decisions. To demonstrate the potential timing benefits of processing ROIs instead of the entire frame, we cropped synchronized images from the nuScenes dataset and fed them into UniAD's feature extraction module to measure the inference time. The UniAD's feature extraction module with ResNet101 and FPN is widely used

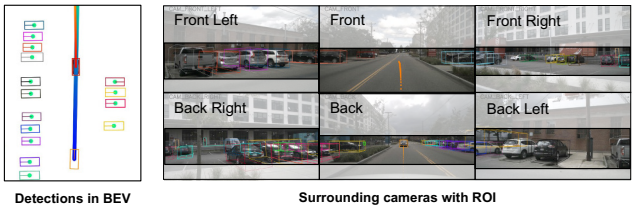

Fig. 2. An example of environment-aware ROI.

in BEV perception models [4], [6], [13], [14], [26], [27]. Here we assume each camera only has one ROI and all ROIs have the same square shape and size. Figure 3 illustrates the inference latency results when ROIs are set at different pixel dimensions with the number of cameras ranging from one to six. Using ROIs can significantly reduce the inference time, with processing time reductions varying between 2x and 35x. Direct cropping of images may cause performance degradation if the cropping area is not carefully designed.

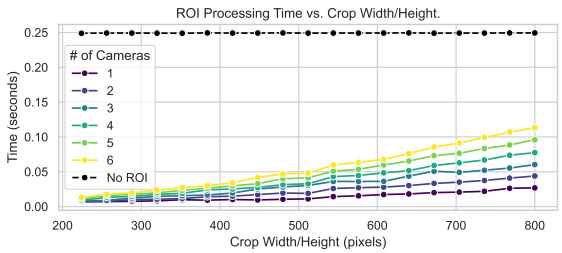

Fig. 3. Potential time savings for ROIs.

ROIs Variability. In practice, the ROIs for each camera often vary in size rather than being uniform. In diverse driving contexts, some cameras might not even be included in the ROI selection. However, DNN inference on GPUs typically requires that tensors within a batch maintain the same shape to optimize throughput [31], [32]. This uniformity is crucial for efficient processing using the Single Instruction, Multiple Thread (SIMT) architecture, which leverages thousands of GPU cores [33]. Consequently, the batch-processing approach is to resize all ROIs to the *smallest* coverage size of them. This

method allows for batching multiple images' ROIs together at the cost of increasing the ROI size, thereby introducing overhead. An alternative approach is to forego batching and process each ROI sequentially for each camera.

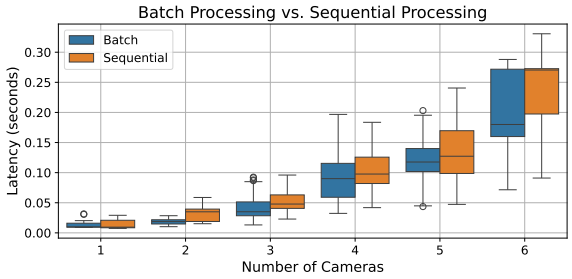

Fig. 4. Comparison of latency for batch and sequential processing with ROIs.

This situation poses an interesting question: *should we process variable-sized ROIs sequentially, or batch process uniformly sized ROIs?* To evaluate the impact of different processing strategies, we compared the inference times of both approaches using UniAD's feature extraction module. Figure 4 presents an inference latency box plot for batch and sequential processing using the same ROIs. We observe that the latency distribution for both processing strategies is quite similar. The latency primarily depends on the ROI sizes for each camera. For ROIs covering just one camera, sequential processing has a slightly lower average latency compared to batch processing. As the number of cameras increases, batch processing demonstrates a smaller average latency. Interestingly, sequential processing can outperform batch processing when ROIs include six cameras if five have small ROIs and one has a large ROI. Consequently, the choice between batch processing and sequential processing requires careful consideration of the ROIs' characteristics.

Insight 1: *ROIs for BEV perception models in autonomous vehicles are environment-aware. This approach can potentially reduce detection delay by decreasing the amount of data that needs to be processed. However, the variable ROI sizes require careful modeling to avoid timing overhead when integrated with BEV perception pipelines on GPUs.*

B. Multi-Camera Data Synchronization

To support dynamic ROI sizes for BEV perception, multi-camera data synchronization is another essential aspect. In AVs equipped with several cameras, the timestamps for each camera that captures the environment can vary greatly [20], [21]. In addition, the BEV perception model requires a batch of synchronized frames, which is usually handled by communication middleware with an approximate time synchronization policy [24], [25]. The actual time difference for synchronized frames could vary from tens to hundreds of milliseconds. Here we discuss the capturing time variations and synchronization time variations separately.

Capturing Time Variations. To show the effect of time discrepancies in data capturing, we use the nuScenes dataset as an example. This dataset includes cameras operating at 12Hz and a LiDAR sensor functioning at 20Hz. In the nuScenes calibration setup, the camera's exposure is triggered when the top LiDAR beam intersects the center of the camera's field of view [20]. As a result, the actual capture times for each camera vary. Figure 5 displays the scatter of maximum time differences for synchronized keyframe images in the nuScenes dataset. Although the keyframes in the nuScenes dataset are largely synchronized, there is still a maximum time deviation of 39ms to 46ms. This time variance is a direct outcome of the 20Hz operation of the LiDAR and the camera's exposure mechanism, which has a theoretical maximum time difference of exactly 50ms. This time difference is fine since the keyframe in the nuScenes dataset is 2Hz and frames are synchronized offline. In practice, AV-equipped cameras operate at 10-30Hz and require online time synchronization. Therefore, SOTA approaches cannot handle it.

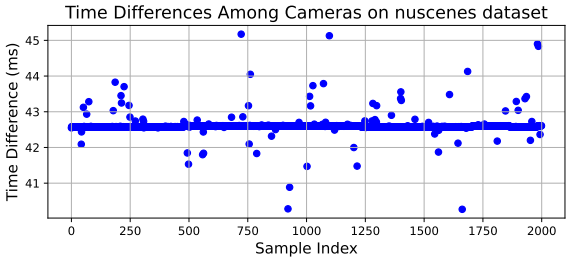

Fig. 5. Time difference on synced nuScenes dataset.

Online Synchronization Time Variations. At runtime, a sweep of image frames is captured and stored in message queues within communication middleware like ROS. ROS synchronizes the cameras' frames using an approximate time method, which tolerates a maximum allowable discrepancy in timestamps between the sensors [24]. For each synchronization cycle, it first determines a pivot, which is the newest image for all cameras, and then generates a candidate group of images from all cameras where the maximum time difference is below the allowable value [25].

This approach can result in dropping images and increasing synchronization delays if one camera's image arrives late. The issue is even more critical for ROI processing, as not all cameras would be included in the ROIs under certain traffic environments and contexts. To illustrate the effect of communication delay on multi-camera image synchronization, we use the nuScenes dataset's sweep data, which includes 12 Hz camera images, and apply ROS `message_filter`'s Approximate Time Policy for synchronization [25]. The maximum allowable time difference is set to 200 ms. Figure 6 shows the synchronization time differences with and without considering capturing time. Without considering capturing time, synchronization time varies from 40 ms to 110 ms, changing gradually because ROS handles multiple topics sequentially. When capturing time is considered, with the message header timestamp set to the actual capturing timestamp, the final synchronization time difference ranges between 40 ms

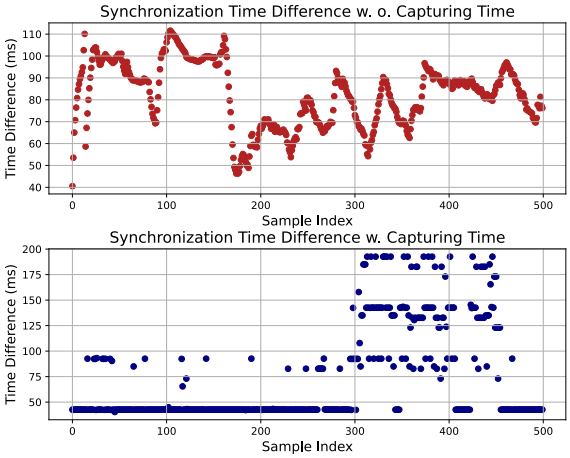

Fig. 6. Synchronization time difference with and without capturing time.

to 200 ms. This range includes both capturing time and ROS communication time. The synchronization time difference changes dramatically, as the camera's capturing time varies across samples. Integrating ROI information with the message synchronization process could potentially reduce the number of message queues, thereby increasing the synchronization success rate.

Insight 2: *Time synchronization is critical to BEV perception. The time difference for synchronized images can reach hundreds of milliseconds due to camera exposure time differences and communication delays. An ROI-aware message synchronizer is essential for achieving real-time BEV perception.*

IV. System Design

This section details the design of RT-BEV. Based on the insights gained in Section III, RT-BEV achieves real-time BEV perception through environment-aware ROI coordinated processing (by addressing Insight 1) and ROI-aware sensor synchronization (by addressing Insight 2). We begin with an overview of RT-BEV, followed by a discussion of the technical challenges and the corresponding research questions. Finally, we introduce the functionality of each component in RT-BEV.

A. System Overview

Figure 7 provides an overview of RT-BEV, a novel framework designed to enhance real-time BEV perception in AVs. The Camera Synchronizer ensures precise time synchronization across multiple sensors ❶, where the synchronization is ROI-aware and TTC-aware to reduce communication delay. Following synchronization, the Feature Split & Merge module processes synchronized camera frames with ROIs ❹. It divides model inference between ROI and non-ROI regions, applying intensive feature extraction to ROI regions via the backbone and neck modules, while non-ROI regions utilize temporal locality for feature generation. The extracted features are then seamlessly integrated. The ROIs Generator supplements this process by creating potential ROI proposals based on detection outputs and contextual data from traffic and the environment ❷, focusing computation on critical areas to drive decisions. Additionally, the Time Predictor enhances system responsiveness by monitoring time-to-collision (TTC) and predicting the inference times for dynamically changing ROI inputs ❸, facilitating preemptive operational adjustments. At the heart of RT-BEV is the Coordinator ❺, which plays a pivotal role in managing keyframe frequency and ROI regions, and in controlling synchronization and ROI processing strategies to minimize e2e latency while ensuring accuracy. Together, these components ensure that the RT-BEV system delivers reliable, real-time data crucial for advanced AV applications.

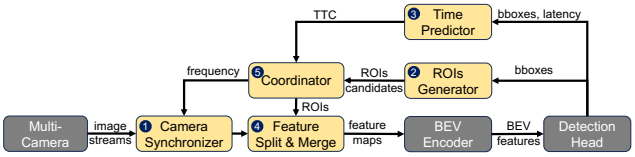

Fig. 7. RT-BEV system overview.

B. Technical Challenges

To support real-time BEV perception, RT-BEV addresses the following challenges:

C1: *Mitigating Synchronization Delay: How to reduce synchronization delays for the multi-camera system?* RT-BEV employs a Camera Synchronizer with a novel approximate time synchronization mechanism that is aware of ROI selection and supports flexible runtime reconfiguration.

C2: *Modeling the Traffic Environment: How to model the traffic environment and driving context to generate ROIs for each camera to maintain detection accuracy?* RT-BEV includes the ROIs Generator that constructs ROI proposals for each camera by leveraging the locality of detected moving objects and view transformation.

C3: *Processing Variable ROIs: How to design a BEV perception pipeline to handle variable ROI sizes efficiently?* RT-BEV uses a Feature Split & Merge module that supports the generation and updating of feature maps with variable-sized ROIs. The Time Predictor also models ROI timing with batching or sequential processing and time-to-collision (TTC).

C4: *Balancing Real-Time and Accuracy: How to trade-off between real-time processing and accuracy to ensure the safety of autonomous vehicles?* RT-BEV includes a Coordinator that collects intermediate results on ROI proposals, TTC, and keyframe frequency to determine the best trade-off between real-time performance and accuracy.

C. Camera Synchronizer

The Camera Synchronizer is designed to support the FlexibleTimeSync policy which has two novel designs: flexible and ROI-aware synchronization queues, and adaptive allowable maximum time difference. Figure 8 provides a schematic comparison of the FlexibleTimeSync in

RT-BEV and ApproximateTimeSync in ROS when handling the same message groups [24], [25]. Both approaches depict a central pivot point coordinating the synchronization of messages from multiple cameras labeled cam1 through cam6. The ApproximateTimeSync policy searches images from all six cameras and synchronizes them based on approximate time. We can find three synchronized groups (T_1, T_2, T_3) with the corresponding time differences (Δt_1, Δt_2, Δt_3). In contrast, FlexibleTimeSync takes an adaptive approach that determines which cameras to synchronize based on ROIs. Therefore, we can observe the FlexibleTimeSync policy only covers three cameras for synchronized groups T_2' and T_3'. Moreover, the time differences ($\Delta t_2'$, $\Delta t_3'$) under FlexibleTimeSync are much smaller than ApproximateTimeSync. Synchronizing with a smaller group reduces the probability of excessive delays.

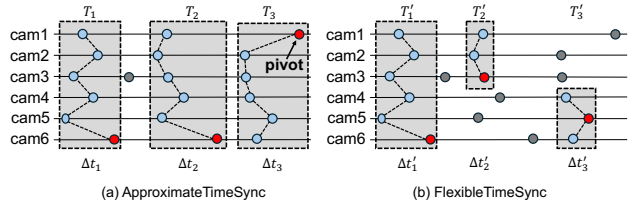

Fig. 8. An illustration of the FlexibaleTimeSync policy in RT-BEV.

Algorithm 1 Camera Synchronization in RT-BEV.

1: **Input:** Topic-specific queues $queues$, keyframe frequency k_f, ROI cameras C, last published set S
2: **Output:** Published set T
3: **function** SYNCHRONIZER($queues$, k_f, C)
4: **for each** cam_topic **in** $queues$ and ROI cameras C **do**
5: Discard messages older than the last message in set S
6: Insert new messages into $queues[cam\_topic]$
7: **end for**
8: Wait until each $queue$ contains at least one message
9: $pivot \leftarrow$ latest message among the first in each $queue$
10: $candidateSet \leftarrow$ messages with time differences from $pivot$ lower than $1/k_f$
11: **while** new messages arrive **do** ▷ Optimize and Publish Set
12: Update $candidateSet$ with new messages
13: **if** a better set is found or end of queue is reached **then**
14: $T \leftarrow candidateSet$
15: Publish T
16: **break**
17: **end if**
18: **end while**
19: **end function**

Algorithm 1 describes the overall process. If some (back, back left, back right) cameras are not included in the ROIs, then the Camera Synchronizer skips these cameras and focuses only on the front three cameras (Lines 4–7). Synchronization with a smaller number of messages increases the possibility of better and faster time alignment. The Camera Synchronizer updates the ROIs and uses them to determine which camera frames to be included in the synchronization at runtime. Furthermore, if the keyframe frequency is low for certain driving contexts (e.g., driving at low speed) then the allowable maximum time difference would be higher, else it would be lower. The Camera Synchronizer takes updates on the time-to-collision (TTC) and adjusts the allowable maximum time difference, consistently with the timing requirements of the traffic environment (Lines 9–10). When new messages arrive, the candidate set is updated (Lines 11–12). When it finds an optimized set or reaches the end of the queue, it assigns this set to T, publishes it then exits the loop (Lines 13–19).

D. ROIs Generator

To ensure real-time and accurate BEV perception, it is essential to select those ROIs that cover critical segments and exclude unnecessary ROIs. There are two key components in the ROIs Generator: (i) a context filter for determining the vehicle's driving state and selecting relevant cameras, and (ii) a runtime traffic environment analyzer for generating ROIs based on BEV perception results.

The context filter identifies the vehicle's driving status (moving forward, turning/changing lanes, or moving backward) and selects the corresponding cameras as ROIs. Figure 9 illustrates the context-related cameras. Moving forward uses the front cameras; turning/changing lanes uses the front and side cameras; moving backward uses the rear cameras. Driving status is primarily based on the global planner and the vehicle's speed, which generates the route from the AV's current position to the destination, just like Google Maps.

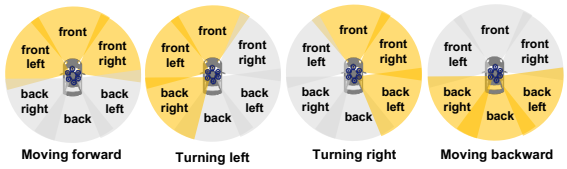

Fig. 9. The context-related cameras for ROIs.

The traffic environment analyzer generates ROIs using BEV perception results from the detected keyframes to guarantee perception accuracy. Figure 10 shows the workflow for generating ROIs. The analyzer processes bounding boxes, scores, and labels to transform BEV data to align with the vehicle's perspective. This involves transformations from BEV to Ego view and then to Sensor view, using LiDAR and camera intrinsics for accurate mapping. The ROIs are refined to ensure minimum 2D coverage in the camera view, crucial for environmental awareness. Algorithm 2 details the ROI generation process. It takes BEV bounding boxes, labels, scores, driving commands, and sample indices as inputs. The context analyzer determines relevant cameras (Line 4), and bounding boxes are filtered based on a score threshold of 0.5, which follows the general setting in Non-Maximum Suppression (NMS) to filter out low-probability objects (Line 5) [34]. For each ROI camera, bounding boxes are transformed and projected to image UV coordinates (Lines 6–10). The algorithm then computes the minimum bounding box covering all boxes (Lines 11–18), updating the ROIs for each camera to ensure precise ROI generation (Lines 19–21).

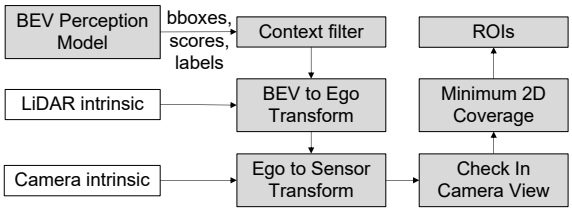

Fig. 10. The ROI generation process.

Algorithm 2 ROI Generation Process for Each Camera

Input: BEV bounding boxes $bboxes$, $labels$, $scores$, ego vehicle driving commands (v_x, v_y)
Output: Updated camera bounding boxes $rois\_box$
function ROIGENERATOR($bboxes$, $labels$, $scores$, v_x, v_y)
  $cam\_rois \leftarrow$ context analyzer with (v_x, v_y)
  Filter $bboxes$ and $labels$ where $scores > 0.5$
  for each $camera$ **in** cam_rois **do**
    $cam\_intrisic \leftarrow$ Calibration data for $camera$
    $boxList$ = [] ▷ initialize empty list for UV boxes
    for each $bbox$ **in** $bboxes$ **do**
      $bbox$ = $bbox$.transform ▷ Transform from BEV to camera's coordinates
      if box_in_image($bbox$, $cam\_intrisic$) **then**
        $bbox \leftarrow bbox$.project($cam\_intrisic$) ▷ project to image UV coordinates
        $boxList$.append($bbox$)
      end if
    end for
    if $boxList$ is not empty **then**
      $bounding\_uv \leftarrow boxList$ ▷ Get minimum box to cover all boxes
    end if
    $rois\_box[camera]$ = $bounding\_uv$
  end for
  return $rois\_box$
end function

E. Time Predictor

The Time Predictor is designed to monitor the timeline of BEV perception and consists of two components: predicting (a) inference time for ROIs and (b) Time-to-Collision (TTC).

The first component of the Time Predictor focuses on calculating the time required to process images for BEV perception, which utilizes dynamic ROIs tailored to each camera and varying in size. For CNN-based models, the inference time has a direct relationship with the ROIs' height h and width w [11], [35], [36]. Here we predict the inference time of ROIs with sequential or batch processing. In sequential processing, the total inference time T_{seq} is the sum of the inference times for each of the six cameras, where $f(h_i, w_i)$ is the inference time for the i-th camera's ROI:

$$T_{seq} = \sum_{i=1}^{6} f(h_i, w_i) \tag{1}$$

In batch processing, the ROIs are resized to a unified height and width, represented by the maximum dimensions among all cameras in ROIs. The total inference time T_{batch} for this unified size is:

$$T_{batch} = f'(b, h', w') \tag{2}$$

b is the batch size, while h' and w' are the minimum height and width to cover all the camera's ROI.

The second component of the Time Predictor estimates the TTC based on the objects detected in the BEV space, which corresponds to the keyframe time interval. This prediction is crucial for proactive vehicle control and safety measures. The TTC calculation utilizes the minimum distance (d_{min}) between the ego vehicle and any detected object along its projected path. Given that the BEV perception bounding boxes are in a 3D space, the distance can be accurately calculated. The TTC is then computed using the vehicle's current speed (v) and the directional commands (d_{cmd}), where TTC is $\frac{d_{\text{min}}}{v \cdot d_{\text{cmd}}}$. A static time ($T_{\text{offset}}$), required for the vehicle's planning and control algorithms to react, is subtracted from this estimated collision time to provide a more accurate prediction of when corrective actions need to be initiated.

F. Feature Split & Merge

With ROIs generated from each camera, it is crucial to apply them to the BEV perception to save inference time without losing accuracy. RT-BEV focuses on feature extraction which is widely used in the vision-centric BEV perception models [4], [6], [13], [14], [26], [27]. The Feature Split & Merge module is designed based on the observation of spatial and temporal locality in streaming perception. Figure 11 illustrates the general process of feature split and merge. First, there is a high similarity between the $(N-1)$-th frame and the N-th frame, indicating high temporal locality. The $(N-1)$-th frame, considered a keyframe, undergoes the whole-frame feature extraction. In contrast, the N-th frame uses an ROI to crop data, which is then processed by a CNN backbone for feature extraction. Comparing the feature maps of the $(N-1)$-th and N-th frames shows that the spatial information is maintained, and the ratio of feature map size and similar areas is consistent with the ROI.

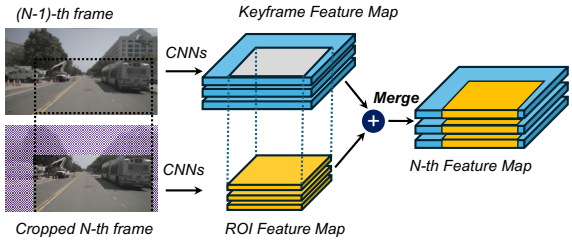

Fig. 11. An illustration of the feature split & merge process.

The spatial and temporal locality in consecutive frames in BEV perception offers a unique advantage for detection accuracy [4], [37]. The feature split & merge approach leverages temporal locality between consecutive frames and spatial locality of ROIs to generate ROI-processing feature maps. Algorithm 3 provides the detailed feature split & merge process. Cameras without ROIs will be skipped (Lines 8–9). Data padding is added to ROI to handle offset between consecutive frames (Lines 11–13). The image will be cropped based on refined ROI and applied to the backbone (Lines 14–15). For areas outside these ROIs, the system uses features

extracted from previous frame analyses to predict the current non-ROI features. The predicted non-ROI features are then seamlessly merged with the freshly processed ROI features to create a unified feature map (Lines 16–21). The size of the feature map in the FPN is halved after each convolution layer, so we adjust the feature map sizes accordingly. This integrated feature map ensures complete scene understanding, enhancing the vehicle's navigation capabilities and overall safety in diverse driving conditions.

Algorithm 3 Image ROIs Split and Merge

1: **Input:** List of images $imgs$, ROIs bounding boxes $rois\_box$, previous keyframe features pre_key_feats
2: **Output:** Features from images $imgs\_feats$
3: **function** ROISSPLITMERGE($imgs, rois\_box, pre\_key\_feats$)
4: $imgs\_feats \leftarrow pre\_key\_feats$
5: **Initilize** divisible size d_s, data padding size $data\_padding$
6: **for each** img **in** $imgs$ **do**
7: $xmin, ymin, roi\_w, roi\_h$ = $rois\_box[img.cam]$
8: **if** roi_w == 0 **or** roi_h == 0 **then**
9: **continue**
10: **else**
11: ▷ Add data padding and make size divisible by d_s
12: $roi\_w = ((roi\_w + data\_padding + d_s - 1)//d_s) * d_s$
13: $roi\_h = ((roi\_h + data\_padding + d_s - 1)//d_s) * d_s$
14: $img\_cropped$ = img[, $ymin$:roi_h, $xmin$:roi_w]
15: $feats \leftarrow$ IMAGEBACKBONE($img\_cropped$)
16: p_x, $p_y \leftarrow xmin$ / 8, $ymin$ / 8
17: **for each** feature map $feat$ from $feats$ **do**
18: h_f, $w_f \leftarrow feat$.size()
19: $imgs\_feats$[..., p_y:$p_y + h_f$, p_x:$p_x + w_f$]
20: p_x, $p_y \leftarrow p_x$ / 2, p_y / 2
21: **end for**
22: **end if**
23: **end for**
24: **return** $imgs\_feats$
25: **end function**

G. Coordinator

The coordinator is the central component in RT-BEV designed to achieve a trade-off between real-time processing and accuracy. It collects all intermediate results, including ROI proposals, predicted times for the ROIs, and time-to-collision (TTC) estimates. The coordinator manages BEV perception with two modes: keyframe and ROI modes. In keyframe mode, the model processes images from all six cameras to generate a complete feature map and detection results. In ROI mode, the model processes only the data within the ROIs.

Algorithm 4 presents the overall coordination process. The $counter$ is initialized outside the function as zero. The frequency of keyframes is determined by the TTC (Line 4). ROIs and their corresponding cameras ($rois$ and $rois\_cam$) are generated by the ROIs Generator based on previous detection results (Line 5). These ROIs, along with the keyframe frequency K_f and $rois\_cam$, are then used for further processing. For non-keyframes, the feature split & merge module is invoked along with the predicted processing times T_{seq} and T_{batch} (Lines 7–8). If the current frame is a keyframe, the entire image is fed into the backbone network. (Lines 9–10) Subsequently, feature maps are processed through the neck, BEV encoder, and detection head to obtain the current detection results $curr\_res$ (Lines 12–14). The $counter$ is reset to zero to indicate a keyframe (Lines 15–17).

Algorithm 4 The Coordination Process

1: **Input:** Image queue $queue$, previous detection results res, time-to-collision TTC, predicted times T_{seq}, T_{batch}, $counter$
2: **Output:** Current detection results $curr\_res$
3: **function** COORDINATOR($img, queue, res, TTC, T_{seq}, T_{batch}$)
4: $K_f = \frac{1}{TTC}$
5: $rois, rois\_cam \leftarrow$ ROIGENERATOR(res)
6: $img \leftarrow$ SYNCHRONIZER($queue$, K_f, $rois\_cam$)
7: **if** $counter \neq 0$ **then**
8: $img\_feats \leftarrow$ ROISSPLITMERGE(img, $rois$, T_{seq}, T_{batch})
9: **else**
10: $img\_feats \leftarrow$ IMAGEBACKBONE(img)
11: **end if**
12: $img\_feats \leftarrow$ IMAGENECK(img_feats)
13: $bev\_feats \leftarrow$ BEVENCODER(img_feats)
14: $curr\_res \leftarrow$ DETECTIONHEAD(bev_feats)
15: **if** $counter$ % $K_f == 0$ **then**
16: $counter$ = 0 ▷ Signal for keyframe
17: **end if**
18: **return** $curr\_res$
19: **end function**

V. IMPLEMENTATION

We integrate RT-BEV's implementation into UniAD's perception pipeline based on ROS in a resource-limited GPU desktop [6], [24]. UniAD's feature extraction with ResNet and FPN is widely used for vision-centric BEV perception models [4], [6], [13], [14], [26], [27].

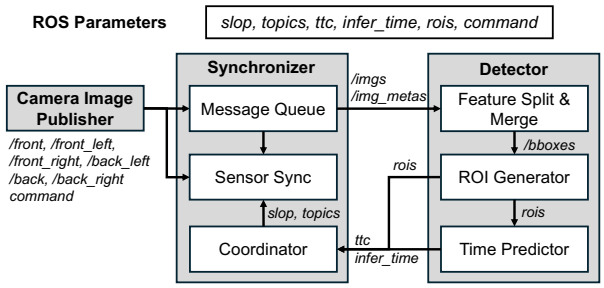

Fig. 12. ROS framework for RT-BEV.

ROS Framework. Based on the BEV perception pipeline illustrated in Figure 1, we developed a ROS framework for its perception system (Figure 12). This framework consists of three ROS nodes (`Publisher`, `Synchronizer`, and `Detector`), nine ROS topics, and six ROS parameters.

TABLE I
COMPARISON OF SYNC POLICIES.

Feature	`ApproximateTimeSync`	`FlexibleTimeSync`
Topics	Fixed topics number and name	reconfigurable through `topics`
Slop	Fixed maximum time difference	reconfigurable through `slop`

The Publisher node publishes the NuScenes dataset across /front, /front_left, /front_right, /back_left, /back, /back_right and the driving command (command), indicating the vehicle's direction. The Synchronizer node includes a message queue, a custom Camera Synchronizer, and a coordinator. The Camera Synchronizer implements FlexibleTimeSync policy which supports runtime reconfiguration of synchronization slop and topics. Table I compares the FlexibleTimeSync policy with the default ApproximateTimeSync policy [25]. The coordinator checks slop and topics every second to determine if reconfiguration is needed. The Detector node processes synchronized image data from /imgs and metadata from /img_metas, applying feature split and merge operations followed by detection. The bounding boxes (/bboxes) generated are sent to the ROIs Generator to create ROIs, shared via the parameter rois. Additionally, the time predictor uses the bounding boxes and ego vehicle velocities from /img_metas to calculate the time-to-collision (ttc) and predicted inference time for batching and sequential processing (infer_time). The coordinator reconfigures the synchronizer's slop based on ttc, updates topics based on rois and command, and adjusts rois based on infer_time.

Baseline. UniAD is used as the baseline for comparison. RT-BEV optimizes feature extraction, making UniAD representative because its feature extraction with ResNet and FPN has been widely used in vision-centric BEV perception, including BEVformor, UniAD, Fast-BEV, and Simple-BEV. The novelty of RT-BEV lies in two main aspects: flexible time synchronization for multi-camera BEV perception data and adaptive ROIs for BEV perception. To demonstrate the effectiveness of these features, we designed four testing cases. Table II describes each testing case.

TABLE II
DESCRIPTIONS OF TESTING CASES

Testing Cases	Descriptions
NR+ATS (SOTA)	No ROIs + ApproximateTimeSync
NR+FTS	No ROIs + FlexibleTimeSync
AR+ATS	Adaptive ROIs + ApproximateTimeSync
AR+FTS (RT-BEV)	Adaptive ROIs + FlexibleTimeSync

VI. EVALUATION

We evaluate the effectiveness of RT-BEV in guaranteeing the real-time and accuracy of BEV perception through comprehensive experiments and ablation studies.

A. Experimental Setup

Hardware and software setup. The GPU desktop has 24 12th Gen Intel® Core™ i9-12900K CPUs with the highest frequency of 3.3GHz. The platform has an NVIDIA GeForce RTX 3080 GPU providing 29.8 teraFLOPS for FP32. The GPU card has 10GB GDDR6 memory. In addition, the platform has 32 GB DDR4 memory. The libraries installed for ML-related applications include CUDA Driver 545.29.06, CUDA runtime 12.3, torch v1.9.1+cu111, torchvision v0.10.1+cu111, mmcv-full v1.4.0, mmdet v2.14.0, and mmdetection3d v0.17.1. ROS Melodic is deployed as the communication middleware.

Dataset. The nuScenes dataset is used as the input to the perception pipeline [20]. The dataset includes 11,628 images with a resolution of 1600x928 covering six cameras and 10 traffic scenarios. Images are published as ROS Image at 12Hz.

Metrics. We evaluate the testing cases using latency, accuracy, and frame efficiency score (FES) as metrics. We also measure communication delay, detection delay, and e2e latency. For accuracy, we assess cosine similarity at the feature map level, mean Average Precision (mAP) for object detection [38], and Average Multi-Object Tracking Precision (AMOTP) for object tracking [20], [39]. AMOTP represents the average localization error of tracked objects across multiple recall levels, where $d_{i,t}$ denotes the position error for object i at time t, and TP_t is the number of true positives at time t. A lower AMOTP indicates better tracking performance.

$$AMOTP = \frac{1}{n-1} \sum_{r \in \left\{\frac{1}{n-1}, \frac{2}{n-1}, \ldots, 1\right\}} \frac{\sum_{i,t} d_{i,t}}{\sum_t TP_t} \tag{3}$$

Additionally, we define the FES, which integrates both latency and accuracy. The frame efficiency score (FES) is a measure of the efficiency of a system by considering its processing time and the effectiveness of synchronization and accuracy, defined as Eq. (4). The average accuracy is calculated as the average mAP over all classes.

$$\text{FES} = \frac{\text{processed frames} \times \text{average accuracy}}{\text{average latency}} \tag{4}$$

B. Latency Reduction

The evaluation of latency reduction includes the communication delay, the detection delay, and the end-to-end latency.

Communication Delay. As presented in Figure 1, the communication delay describes the time from camera exposure until the synchronized images are fed into the BEV perception pipeline. Table III shows the synchronization and communication delay. The communication delay is defined as the timestamp difference between two consecutive synchronized images. This includes both the sync delay and queuing delay, reflecting the predictability of the synchronization process. From Table III, we can observe that RT-BEV reduces the average sync delay from 72.4ms to 64.9ms. Regarding the communication delay, RT-BEV reduces it from 151.7ms to 110.7ms on average (a 1.4× speedup). This reduction is primarily due to the introduction of FTS (FlexibleTimeSync). FTS also helps to reduce the maximum (worst-case) communication delay. NR+FTS and RT-BEV show maximum communication delays of 251.4ms and 288ms, respectively, which is a significant improvement (48×-55× speedup) compared to NR+ATS (14071.3ms). This substantial communication delay

TABLE III
COMPARISON OF SYNC DELAY AND TIMESTAMP DELTA

Cases	Sync Delay (ms)	Communication Delay (ms)	Min Communication Delay (ms)	Max Communication Delay (ms)
NR+ATS	72.4	151.7	103.7	14071.3
NR+FTS	71.3	**110.7**	50.2	**251.4**
AR+ATS	65.4	131.5	104.3	15251.8
RT-BEV	**64.9**	**110.7**	**23.7**	288.0

in NR+ATS occurs when there is a high time difference between queues, resulting in no candidate for synchronization. The reconfigurable design in the FTS policy addresses this issue by adjusting the number of synchronization topics and time difference slops, increasing the likelihood of successful synchronization. The Camera Synchronizer shows effective performance in reducing worst-case communication delay and it makes the synchronization process more predictable.

Detection Delay. The detection delay covers the time from receiving synchronized images until the detection process is complete. Therefore, RT-BEV also includes the ROIs generator, time predictor, and feature split & merge in the detection delay. Figure 13 shows the boxplot for detection delay under four cases. We can observe that the introduction of adaptive ROIs significantly decreases the detection delay, although some outliers exist due to keyframe detection. Tables IV and V show detailed comparisons of the average, minimum, and maximum delays for feature extraction and detection.

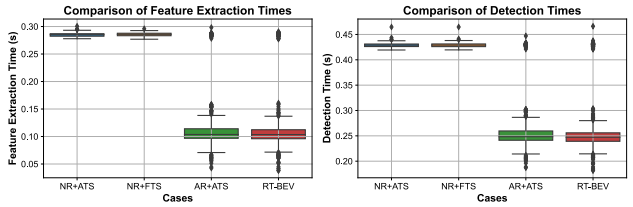

Fig. 13. Boxplot of feature extraction and detection latency.

For feature extraction, adaptive ROIs help reduce the latency from approximately 285ms to 123ms, achieving a 2.3× speedup. Regarding the minimum latency, RT-BEV and AR+ATS show a 6.5-7.3× speedup since RT-BEV processes smaller ROIs rather than the entire frame when there is high locality. In terms of detection delay, which includes feature extraction and additional overhead from generating ROIs and conducting time prediction, RT-BEV and AR+ATS still show much lower latency with around a 1.6× speedup. The introduction of adaptive ROIs is highly effective in reducing detection latency based on the similarity of consecutive images.

End-to-end latency. As for the end-to-end latency which covers both communication and detection. Table VI shows the e2e latency for four cases. We can observe that RT-BEV shows 1.5× speedup in average e2e latency, 2.6× speedup in minimum e2e latency, and 19.3× speedup in maximum e2e latency. Overall, the FTS reduces the communication delay while the adaptive ROIs reduce the detection delay.

TABLE IV
COMPARISON OF FEATURE EXTRACTION TIME METRICS

Cases	Average (ms)	Speedup	Min (ms)	Speedup	Max (ms)	Speedup
NR+ATS	284.9	-	277.8	-	300.5	-
NR+FTS	285.4	1.0×	276.9	1.0x	**295.9**	1.0×
AR+ATS	125.1	**2.3**×	42.6	6.5×	298.3	1.0×
RT-BEV	**123.0**	**2.3**×	**38.1**	**7.3**×	**290.3**	1.0×

TABLE V
COMPARISON OF DETECTION DELAY

Cases	Average (ms)	Speedup	Min (ms)	Speedup	Max (ms)	Speedup
NR+ATS	429.0	-	419.3	-	464.7	-
NR+FTS	428.7	1.0×	419.6	1.0×	**464.7**	1.0×
AR+ATS	**269.7**	1.6x	187.1	2.2×	447.1	1.0×
RT-BEV	**266.9**	**1.6**×	**181.5**	**2.3**×	466.2	1.0×

TABLE VI
COMPARISON OF END-TO-END LATENCY

Cases	Average (ms)	Speedup	Min (ms)	Speedup	Max (ms)	Speedup
NR+ATS	580.6	-	523.0	-	14536.0	-
NR+FTS	539.4	1.1×	469.8	1.1×	716.1	**20.3x**
AR+ATS	401.2	1.5×	291.4	1.8×	15698.8	0.9×
RT-BEV	**377.6**	**1.5x**	**205.2**	**2.6**×	754.2	19.3×

C. Detection Accuracy

In addition to latency, detection accuracy is an essential aspect. We mainly discuss the evaluation of detection accuracy in three aspects: the number of synchronized and processed frames, feature map correctness, and accuracy for object detection and tracking.

Synchronized and Processed Frames. When integrating the BEV perception into a ROS-based pipeline, not every frame from the camera is processed by the detector due to the communication middleware, which only filters synchronized data. Therefore, there is a difference in the number of synchronized and processed frames under the four cases. This frame number difference indeed impacts detection accuracy. Table VII compares the number of synchronized and processed frames. We observe that the default ATS policy only synchronizes 70-80% of frames, and the BEV perception model processes 14-22% of frames. In contrast, the FTS improves the synchronization percentage to 97%, and RT-BEV processes 26 % of frames, which is a 1.8x speedup compared with NR+ATS. This improvement is mainly due to the ROIs-aware reconfiguration of the synchronization process, which allows more frames to get synchronized.

Feature Maps Correctness. To evaluate the correctness of object detection at the feature map level. We utilize the Feature Split & Merge to process images with ROIs, yielding a feature map $\mathbf{A}$. For comparison, we apply model inference to whole images to obtain feature map $\mathbf{B}$. The similarity between

TABLE VII
COMPARISON OF SYNC FRAMES AND PROCESSED FRAMES

Cases	Sync Frames	Sync Ratio (%)	Processed Frames	Processed Ratio (%)
NR+ATS	1375	70.9 (–)	275	14.2 (–)
NR+FTS	**1886**	**97.3 (1.4×)**	356	18.4 (1.3×)
AR+ATS	1569	81.0 (1.1×)	432	22.3 (1.6×)
RT-BEV	**1886**	**97.3 (1.4×)**	**508**	**26.2 (1.8×)**

feature representations derived from the backbone model is assessed using cosine similarity. Figure 14 presents the cosine similarity comparison of whole-frames and ROIs processing.

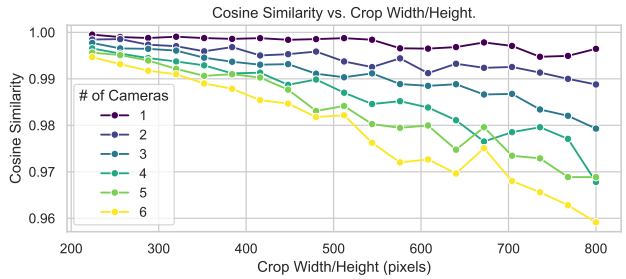

Fig. 14. Cosine similarity for original and ROI-based feature maps.

High cosine similarity scores are observed even when the ROI size is 800x800 (half of the original size), indicating high locality. This is because there are limited and predictable movements of pixels in high FPS camera data, enhancing the efficiency of ROI-based processing. This result demonstrates the locality in consecutive frames for autonomous driving and reflects the effectiveness of feature split & merge.

Object Detection and Tracking. The final step is to evaluate the performance of BEV perception in object detection and tracking. To demonstrate the effectiveness of the proposed ROIs processing compared with processing the whole frame, we conduct an offline test where every synchronized frame is fed into the BEV perception pipeline. We collect the final results to calculate the mAP and AMOTP. Another baseline that applies static ROIs is also implemented for comparison. Figure 15 shows the results under NR-ATS and RT-BEV.

For object detection, we observe that RT-BEV shows comparable performance to NR+ATS, with slightly better performance in detecting trucks, buses, and traffic cones, which are relatively large and static objects. The average mAP for NR+ATS and RT-BEV is almost the same. Regarding object tracking, static ROIs show much better performance than both RT-BEV and NR+ATS. The potential reason is that fixed ROIs maintain spatial information while filtering out unrelated objects for tracking. On average, RT-BEV and NR+ATS show comparable AMOTP.

D. Frame Efficiency Score

FES is designed to show the combined performance in latency and accuracy. Therefore, we collect the average e2e

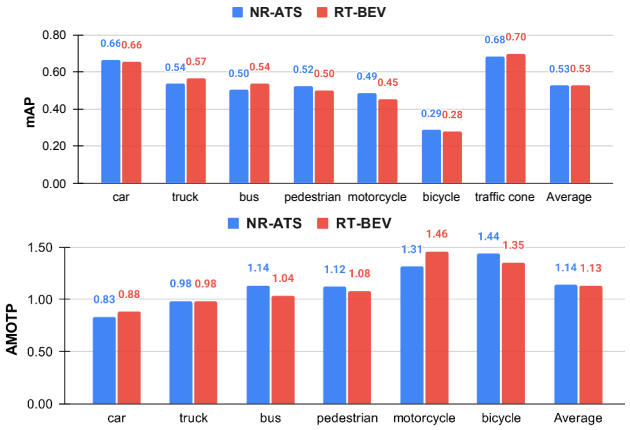

Fig. 15. mAP for object detection and AMOTP for object tracking.

latency, processed frames, and average detection accuracy to compare FES. The results are shown in Table VIII. As a comparison, we found that RT-BEV achieves lower e2e latency, a much better number of processed frames, and a comparable detection accuracy. The final FES of RT-BEV achieves 2.9× speedup compared with the NR+ATS.

This significant speedup demonstrates RT-BEV's efficiency in handling computational resources and optimizing the processing pipeline. By increasing processed frames, maintaining high accuracy, and reducing latency, RT-BEV greatly enhances the real-time performance crucial for AV operations.

TABLE VIII
COMPARISON OF VARIOUS METRICS

Metrics	NR-ATS	NR-FTS	AR-ATS	RT-BEV
Average e2e latency (ms)	580.6	539.4	401.2	**377.6**
Processed Frames	275	356	432	**508**
Average accuracy (%)	0.369	0.369	0.37	**0.37**
FES (speedup)	0.175 (–)	0.244 (1.4×)	0.398 (2.3×)	**0.498 (2.9×)**

E. Alternative Heuristics for ROI Generation

Besides RT-BEV's method for generating ROIs, we have implemented and evaluated alternative heuristics, including static, dynamic, and unified ROIs. For static ROIs, batch processing is applied to a fixed region of each image, which is set to cover half of the image in our implementation. In contrast, dynamic ROIs are designed with variable sizes for each camera and sequential processing is used to handle dynamically changing regions. The unified ROI approach resizes the generated ROIs and conducts batch processing.

To assess the impact of these ROI-generation strategies on perception accuracy, we have used the same camera frames across all approaches and collected results for both object detection and tracking. Tables IX and X present the corresponding mAP and AMOTP metrics for each class. The "Base" strategy refers to using the entire image for perception. For object detection, the base strategy is observed to outperform all the others, while the dynamic ROI approach exhibits

TABLE IX
COMPARISON OF MAP FOR DIFFERENT ROI-GENERATION STRATEGIES

mAP	Unified ROIs	Dynamic ROIs	Static ROIs	Base
car	0.636	0.605	0.578	0.665
truck	0.492	0.473	0.365	0.535
bus	0.554	0.487	0.360	0.503
pedestrian	0.519	0.390	0.420	0.524
motorcycle	0.462	0.426	0.422	0.487
bicycle	0.270	0.305	0.418	0.287
traffic_cone	0.564	0.122	0.578	0.685
Average	**0.499**	**0.401**	**0.449**	**0.527**

TABLE X
COMPARISON OF AMOTP FOR DIFFERENT ROI-GENERATION STRATEGIES

AMOTP	Unified ROIs	Dynamic ROIs	Static ROIs	Base
bicycle	1.427	1.583	1.479	1.436
bus	1.050	1.134	1.464	1.135
car	0.873	0.928	0.991	0.833
motorcycle	1.449	1.492	1.383	1.313
pedestrian	1.148	1.354	1.243	1.120
truck	1.067	1.087	1.316	0.984
Average	**1.169**	**1.263**	**1.313**	**1.137**

a significant drop in performance. This is likely due to its reliance on previous frames for ROI generation, where missed detections in one frame can result in missed regions containing objects in subsequent frames. For object tracking precision, the "Base" strategy again demonstrates the best performance, while the static ROIs strategy shows the worst.

VII. DISCUSSION

A. *Generality of* RT-BEV

RT-BEV is designed as a versatile solution for multi-camera, end-to-end BEV perception, encompassing both communication and detection tasks. The proposed flexible time-synchronization policy is built on top of ROS, making it applicable to *any* ROS-based system, and can also be extended to ROS2 with minimal effort.

For the detection task, RT-BEV is based on UniAD, the state-of-the-art vision-centric BEV perception. RT-BEV enhances UniAD's feature extraction, making RT-BEV a representative solution. Its use of ResNet and FPN for feature extraction is common in vision-centric BEV perception models, such as BEVFormer, UniAD, Fast-BEV, and Simple-BEV.

B. *How Does* RT-BEV *Guarantee Safety?*

To guarantee the safety of AVs, RT-BEV's ROI generation is designed to be lightweight, taking less than 30ms. When image frames are generated and processed at 12–30Hz, the actual movement of objects between two consecutive frames is limited and predictable by the law of physics, allowing us to perceive an object's location differences and movements. We also introduced a TTC model that predicts potential collisions. When TTC decreases, RT-BEV adds keyframes from all cameras into the perception pipeline. Our design ensures critical information is not missed, and the system can respond to potential hazards timelily and effectively.

VIII. RELATED WORK

Vision-centric BEV perception has been widely studied for its unique ability to provide a comprehensive view that aids vehicle navigation [1], [2], [14], [15]. BEV perception captures essential details for navigation and decision-making, thereby enhancing AV navigation and decision-making processes [6]. The accuracy of BEV perception models has significantly improved with the use of vision transformers and cross-attention across multiple cameras and time [4], [5], [40].

However, achieving real-time end-to-end (e2e) BEV perception remains a significant challenge due to the computational demands of high-resolution cameras [12]. Previous work has either designed more lightweight model structures or leveraged model compression to trade-off accuracy for real-time performance [13], [15], [17]. For instance, SparseBEV employs a sparse model design to accelerate inference speed, while SparseViT applies different pruning ratios for regions of varying importance [15], [17]. These approaches, however, can lead to accuracy degradation and overlook the communication delays introduced by the multi-camera system, which are significant in practice [20], [21]. Alternatively, researchers have studied communication middleware to bound worst-case communication delays [22], [23]. However, these methods also fall short of achieving real-time e2e BEV perception due to the lack of integration with detection characteristics. For real-time e2e BEV perception, both communication and computation need to be co-optimized.

IX. CONCLUSION

In this paper, we have presented RT-BEV, a novel framework that co-designs communication and computation to enhance real-time e2e BEV perception tailored for autonomous vehicles. RT-BEV leverages dynamic ROI processing and enhanced system coordination to optimize computation and communication resources to reduce latency and maintain detection accuracy. Comprehensive evaluations using the NuScenes dataset have shown RT-BEV to reduce e2e latency by 1.5× and the number of processed frames by nearly 2× while maintaining high mAP and AMOTP. Furthermore, RT-BEV reduces the worst-case e2e latency by 19.3× compared to SOTA approaches, corroborating its effectiveness in enhancing real-time e2e BEV perception.

ACKNOWLEDGMENTS

This work was supported in part by the Office of Naval Research under Grant No. N00014-22-1-2622. Jinkyu Lee's sabbatical with the University of Michigan was supported in part by the Korean National Research Foundation (NRF) grant funded by the Korea government (MSIT) (2022R1A4A3018824, RS-2024-00438248).

REFERENCES

[1] H. Li, C. Sima, J. Dai, W. Wang, L. Lu, H. Wang, J. Zeng, Z. Li, J. Yang, H. Deng, H. Tian, E. Xie, J. Xie, L. Chen, T. Li, Y. Li, Y. Gao, X. Jia, S. Liu, J. Shi, D. Lin, and Y. Qiao, "Delving into the devils of bird's-eye-view perception: A review, evaluation and recipe," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 46, no. 4, pp. 2151–2170, 2024.

[2] Y. Ma, T. Wang, X. Bai, H. Yang, Y. Hou, Y. Wang, Y. Qiao, R. Yang, D. Manocha, and X. Zhu, "Vision-centric bev perception: A survey," 2023.

[3] S. Liu, L. Liu, J. Tang, B. Yu, Y. Wang, and W. Shi, "Edge computing for autonomous driving: Opportunities and challenges," *Proceedings of the IEEE*, vol. 107, no. 8, pp. 1697–1716, 2019.

[4] Z. Li, W. Wang, H. Li, E. Xie, C. Sima, T. Lu, Y. Qiao, and J. Dai, "Bevformer: Learning bird's-eye-view representation from multi-camera images via spatiotemporal transformers," in *European conference on computer vision*. Springer, 2022, pp. 1–18.

[5] C. Yang, Y. Chen, H. Tian, C. Tao, X. Zhu, Z. Zhang, G. Huang, H. Li, Y. Qiao, L. Lu *et al.*, "Bevformer v2: Adapting modern image backbones to bird's-eye-view recognition via perspective supervision," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2023, pp. 17 830–17 839.

[6] Y. Hu, J. Yang, L. Chen, K. Li, C. Sima, X. Zhu, S. Chai, S. Du, T. Lin, W. Wang *et al.*, "Planning-oriented autonomous driving," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2023, pp. 17 853–17 862.

[7] E. Yurtsever, J. Lambert, A. Carballo, and K. Takeda, "A survey of autonomous driving: Common practices and emerging technologies," *IEEE access*, vol. 8, pp. 58 443–58 469, 2020.

[8] L. Liu, S. Lu, R. Zhong, B. Wu, Y. Yao, Q. Zhang, and W. Shi, "Computing systems for autonomous driving: State of the art and challenges," *IEEE Internet of Things Journal*, vol. 8, no. 8, pp. 6469–6486, 2020.

[9] D. Kang, S. Lee, H. S. Chwa, S.-H. Bae, C. M. Kang, J. Lee, and H. Baek, "Rt-mot: Confidence-aware real-time scheduling framework for multi-object tracking tasks," in *2022 IEEE Real-Time Systems Symposium (RTSS)*. IEEE, 2022, pp. 318–330.

[10] S. Kato, E. Takeuchi, Y. Ishiguro, Y. Ninomiya, K. Takeda, and T. Hamada, "An open approach to autonomous vehicles," *IEEE Micro*, vol. 35, no. 6, pp. 60–68, 2015.

[11] L. Liu, Z. Dong, Y. Wang, and W. Shi, "Prophet: Realizing a predictable real-time perception pipeline for autonomous vehicles," in *2022 IEEE Real-Time Systems Symposium (RTSS)*. IEEE, 2022, pp. 305–317.

[12] Z. Liu, H. Tang, A. Amini, X. Yang, H. Mao, D. L. Rus, and S. Han, "Bevfusion: Multi-task multi-sensor fusion with unified bird's-eye view representation," in *2023 IEEE international conference on robotics and automation (ICRA)*. IEEE, 2023, pp. 2774–2781.

[13] Y. Li, B. Huang, Z. Chen, Y. Cui, F. Liang, M. Shen, F. Liu, E. Xie, L. Sheng, W. Ouyang *et al.*, "Fast-bev: A fast and strong bird's-eye view perception baseline," *arXiv preprint arXiv:2301.12511*, 2023.

[14] A. W. Harley, Z. Fang, J. Li, R. Ambrus, and K. Fragkiadaki, "Simple-bev: What really matters for multi-sensor bev perception?" in *2023 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2023, pp. 2759–2765.

[15] H. Liu, Y. Teng, T. Lu, H. Wang, and L. Wang, "Sparsebev: High-performance sparse 3d object detection from multi-camera videos," in *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 2023, pp. 18 580–18 590.

[16] Y. Zhang, Z. Dong, H. Yang, M. Lu, C.-C. Tseng, Y. Du, K. Keutzer, L. Du, and S. Zhang, "Qd-bev: quantization-aware view-guided distillation for multi-view 3d object detection," in *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 2023, pp. 3825–3835.

[17] X. Chen, Z. Liu, H. Tang, L. Yi, H. Zhao, and S. Han, "Sparsevit: Revisiting activation sparsity for efficient high-resolution vision transformer," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2023, pp. 2061–2070.

[18] F. Dellinger, T. Boulay, D. M. Barrenechea, S. El-Hachimi, I. Leang, and F. Bürger, "Multi-task network pruning and embedded optimization for real-time deployment in adas," *arXiv preprint arXiv:2101.07831*, 2021.

[19] J. Li, Y. Zhao, L. Gao, and F. Cui, "Compression of yolov3 via block-wise and channel-wise pruning for real-time and complicated autonomous driving environment sensing applications," in *2020 25th International Conference on Pattern Recognition (ICPR)*. IEEE, 2021, pp. 5107–5114.

[20] H. Caesar, V. Bankiti, A. H. Lang, S. Vora, V. E. Liong, Q. Xu, A. Krishnan, Y. Pan, G. Baldan, and O. Beijbom, "nuscenes: A multimodal dataset for autonomous driving," in *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, 2020, pp. 11 621–11 631.

[21] P. Sun, H. Kretzschmar, X. Dotiwalla, A. Chouard, V. Patnaik, P. Tsui, J. Guo, Y. Zhou, Y. Chai, B. Caine *et al.*, "Scalability in perception for autonomous driving: Waymo open dataset," in *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, 2020, pp. 2446–2454.

[22] R. Li, X. Jiang, Z. Dong, J.-M. Wu, C. J. Xue, and N. Guan, "Worst-case latency analysis of message synchronization in ros," in *2023 IEEE Real-Time Systems Symposium (RTSS)*. IEEE, 2023, pp. 185–197.

[23] J. Sun, T. Wang, Y. Li, N. Guan, Z. Guo, and G. Tan, "Seam: An optimal message synchronizer in ros with well-bounded time disparity," in *2023 IEEE Real-Time Systems Symposium (RTSS)*. IEEE, 2023, pp. 172–184.

[24] M. Quigley, K. Conley, B. Gerkey, J. Faust, T. Foote, J. Leibs, R. Wheeler, and A. Y. Ng, "ROS: an open-source robot operating system," in *ICRA workshop on open source software*, vol. 3, no. 3.2. Kobe, Japan, 2009, p. 5.

[25] "message_filters," http://wiki.ros.org/message_filters.

[26] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016, pp. 770–778.

[27] T.-Y. Lin, P. Dollár, R. Girshick, K. He, B. Hariharan, and S. Belongie, "Feature pyramid networks for object detection," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2017, pp. 2117–2125.

[28] R. M. Haralick and L. G. Shapiro, "Glossary of computer vision terms." *Pattern Recognit.*, vol. 24, no. 1, pp. 69–93, 1991.

[29] W. Kang, S. Chung, J. Y. Kim, Y. Lee, K. Lee, J. Lee, K. G. Shin, and H. S. Chwa, "Dnn-sam: Split-and-merge dnn execution for real-time object detection," in *2022 IEEE 28th Real-Time and Embedded Technology and Applications Symposium (RTAS)*. IEEE, 2022, pp. 160–172.

[30] W. Luo, Y. Li, R. Urtasun, and R. Zemel, "Understanding the effective receptive field in deep convolutional neural networks," *Advances in neural information processing systems*, vol. 29, 2016.

[31] J. D. Owens, M. Houston, D. Luebke, S. Green, J. E. Stone, and J. C. Phillips, "Gpu computing," *Proceedings of the IEEE*, vol. 96, no. 5, pp. 879–899, 2008.

[32] J. Nickolls and W. J. Dally, "The gpu computing era," *IEEE micro*, vol. 30, no. 2, pp. 56–69, 2010.

[33] J. Sanders and E. Kandrot, *CUDA by example: an introduction to general-purpose GPU programming*. Addison-Wesley Professional, 2010.

[34] P. Developers, "Faster r-cnn model in torchvision," https://github.com/pytorch/vision/blob/33e47d88265b2d57c2644aad1425be4fccd64605/torchvision/models/detection/faster_rcnn.py#L194, 2024, accessed: 2024-09-10.

[35] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," *Advances in neural information processing systems*, vol. 25, 2012.

[36] Y.-H. Chen, T. Krishna, J. S. Emer, and V. Sze, "Eyeriss: An energy-efficient reconfigurable accelerator for deep convolutional neural networks," *IEEE journal of solid-state circuits*, vol. 52, no. 1, pp. 127–138, 2016.

[37] M. Xu, M. Zhu, Y. Liu, F. X. Lin, and X. Liu, "Deepcache: Principled cache for mobile deep vision," in *Proceedings of the 24th annual international conference on mobile computing and networking*, 2018, pp. 129–144.

[38] S. Ren, K. He, R. Girshick, and J. Sun, "Faster r-cnn: Towards real-time object detection with region proposal networks," *Advances in neural information processing systems*, vol. 28, 2015.

[39] X. Zhou, V. Koltun, and P. Krähenbühl, "Tracking objects as points," in *European conference on computer vision*. Springer, 2020, pp. 474–490.

[40] A. Dosovitskiy, L. Beyer, A. Kolesnikov, D. Weissenborn, X. Zhai, T. Unterthiner, M. Dehghani, M. Minderer, G. Heigold, S. Gelly *et al.*, "An image is worth 16x16 words: Transformers for image recognition at scale," *arXiv preprint arXiv:2010.11929*, 2020.