International Journal of Intelligent Robotics and Applications (2024) 8:1038–1056
https://doi.org/10.1007/s41315-024-00363-w

REGULAR PAPER

Towards real-time embodied AI agent: a bionic visual encoding framework for mobile robotics

Xueyu Hou[1] · **Yongjie Guan**[1] · **Tao Han**[2] · **Cong Wang**[2]

Received: 24 April 2024 / Accepted: 26 July 2024 / Published online: 10 August 2024

Abstract
Embodied artificial intelligence (AI) agents, which navigate and interact with their environment using sensors and actuators, are being applied for mobile robotic platforms with limited computing power, such as autonomous vehicles, drones, and humanoid robots. These systems make decisions through environmental perception from deep neural network (DNN)-based visual encoders. However, the constrained computational resources and the large amounts of visual data to be processed can create bottlenecks, such as taking almost 300 milliseconds per decision on an embedded GPU board (Jetson Xavier). Existing DNN acceleration methods need model retraining and can still reduce accuracy. To address these challenges, our paper introduces a bionic visual encoder framework, *Robye*, to support real-time requirements of embodied AI agents. The proposed framework complements existing DNN acceleration techniques. Specifically, we integrate motion data to identify overlapping areas between consecutive frames, which reduces DNN workload by propagating encoding results. We bifurcate processing into high-resolution for task-critical areas and low-resolution for less-significant regions. This dual-resolution approach allows us to maintain task performance while lowering the overall computational demands. We evaluate *Robye* across three robotic scenarios: autonomous driving, vision-and-language navigation, and drone navigation, using various DNN models and mobile platforms. *Robye* outperforms baselines in speed (1.2–3.3 ×), performance (+4% to +29%), and power consumption (−36% to −47%).

Keywords Mobile robotics · Visual encoding · Embodied AI · Computer vision

1 Introduction

Embodied AI agents are designed for interaction and navigation in physical environments. They combine sensory inputs and actuator outputs for perception and action. These systems are deployed on mobile robotic platforms for tasks such as autonomous navigation and object manipulation, in which visual perception (encoding) plays a crucial role (Hu et al. 2019; Qi et al. 2020; Schumann and Riezler 2022; Thomason et al. 2018; Zhu et al. 2021). This visual encoding enables the embodied AI agents to analyze and interpret visual data for decision-making. As shown in Fig. 1, this mimics human-environment interaction, aiding in informed decision-making and appropriate environmental responses in robots. However, the expanding capabilities of vision-based robotic systems also require increased computational capacity, especially in mobile robotics where the embodied AI agents directly interact with their environment. Figure 2 demonstrates this in three major mobile robotic applications of autonomous driving, vision-and-language navigation (VLN), and drone navigation. We measure their latency and power consumption on an embedded GPU platform, Jetson Xavier (https://www.nvidia.com/en-us/autonomous-machines/embedded-systems/jetson-agx-xavier/). For visual encoding, we use a Mono3D object detection model (Chen et al. 2016). We compress the model with state-of-the-art acceleration techniques (Murti et al. 2022), which reduces

✉ Xueyu Hou
xueyu.hou@maine.edu

Yongjie Guan
yongjie.guan@maine.edu

Tao Han
tao.han@njit.edu

Cong Wang
cong.wang@njit.edu

[1] ECE Department, University of Maine, Orono, USA

[2] ECE Department, New Jersey Institute of Technology, Newark, USA

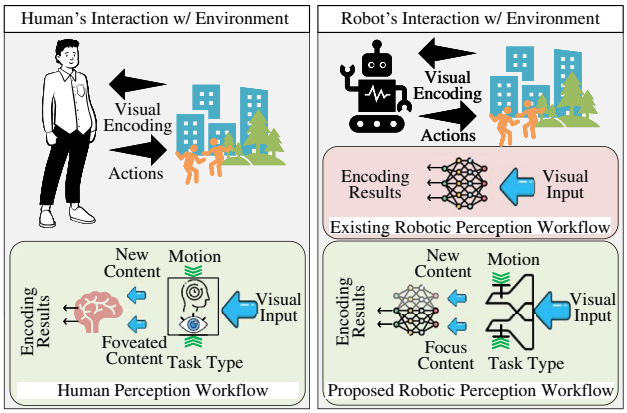


Fig. 1 Interaction with environment: human vs robots

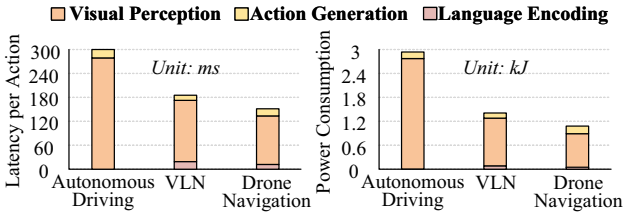


Fig. 2 Latency and power consumption (10 min) measured on Jetson Xavier (https://www.nvidia.com/en-us/autonomous-machines/embedded-systems/jetson-agx-xavier/)

latency by 44.7% with accuracy loss of 8.2%. For action generation, we implement an LSTM model with 512 hidden units (Anderson et al. 2018). For language encoding, we use a standard attentive encoder–decoder model (Bahdanau et al. 2014; Zhang et al. 2020).[1] We observe that, across all the three applications, *even with state-of-the-art model acceleration techniques,* visual encoding accounts for over 70–94% of the latency and power consumption, indicating that it is critical to address the computational load imposed by visual encoding in mobile robotic applications.

Neuroscience has advanced our understanding of human visual encoding and its role in behavior. Key findings include: *First*, the human visual system has a dichotomous mechanism and processes retinal inputs selectively, focusing on high-acuity focus vision in the central field and broader, less detailed encoding in the peripheral field (Stewart et al. 2020). *Second*, iconic memory plays a crucial role in visual encoding, acting as a short-term repository for visual stimuli like shape, color, and motion, and enabling efficient integration of visual information in dynamic scenes (Becker et al. 2000; Coltheart 1980; Gegenfurtner and Sperling 1993). *Third*, the human visual system combines visual inputs with motion information of proprioceptive and vestibular signals for a cohesive representation of the external world, enhancing spatial perception through multisensory integration in the visual and vestibular cortexes (Beers et al. 1996; Gr and Gr 1972).

Inspired by human visual system principles, we propose *Robye*, a bionic visual encoding framework for mobile robotics. As illustrated in Fig. 1, *Robye* integrates the mechanisms of dichotomous processing and motion-aware iconic memory into visual encoding, to reduce computational needs. Specifically, the framework includes: (1) Dual workflows: *Robye* applies high-resolution encoding on task-related focus areas and low-resolution encoding on the rest contents, to concentrate computation to key areas. (2) Motion fusion: Instead of relying on visual information only, *Robye* utilizes motion data (the position, direction, and speed of the robot and surrounding objects) of sensors on mobile robotics to aid in matching contents across frames. (3) Encoding result sharing: *Robye* caches the encoding result of the preceding frame and shares it with the matching contents in the current frame, which reduces computing load of deep neural network (DNN) encoding. The contributions of this work are:

- *Robye* is an efficient visual encoding framework designed for mobile robotics. The innovations to reduce computational costs and keep high performance include: the dichotomous processing for environmental understanding and the motion-based spatial geometric projection for lightweight cross-frame content correlation.
- To localize overlapping contents between frames, we propose to employ sensor-based motion data, which utilizes lightweight, geometric projection-based localization that demands fewer computational resources than image processing methods.
- To reduce DNN computation, we propose to emulate the iconic memory mechanism from human vision, which integrates past encoding results for current frame analysis, and adaptively regulating the visual encoding process based on inter-frame motion changes, to further reduce computational load.
- To preserve the performance of robotic applications, we propose a dual workflow of visual encoding, which mirrors human vision mechanism. It includes task-related focus localization and high-resolution encoding, to reduce computational cost without sacrificing performance of mobile robotic applications.
- We evaluate *Robye* across three mobile robotic applications (autonomous driving, vision-and-language navigation, and drone navigation) using various deep learning models on mobile and embedded devices of Jetson Nano, Jetson Xavier, Google Pixel 7, and Samsung Galaxy S22. The results show that *Robye* outperforms baselines in

[1] In the cases of VLN and drone navigation, the embodied AI agent is given only one natural language instruction at the beginning of the navigation process.

terms of latency (by 1.2–3.3× speedup), performance (by +4% to +29%), and power consumption (−36% to −47%).

2 Background

2.1 Vision-based mobile robotic systems

Mobile robotic systems increasingly use visual encoding in applications such as autonomous driving, vision-and-language navigation (VLN), and drone navigation. This technology is vital for interpreting road scenes in autonomous driving (Shao et al. 2023; Zhang et al. 2021), integrating visual perception with natural language in VLN (Hu et al. 2019), and enabling drones to navigate and avoid obstacles in real time (Blukis et al. 2018). Such integration enhances the cognitive abilities of mobile robots, boosting their autonomy in complex environments (Gu et al. 2022). In these systems, environmental data from sensors and cameras are processed using deep learning, particularly convolutional neural networks (CNNs) (Wijmans et al. 2019b; Ye et al. 2021; Wahid et al. 2021; Anderson et al. 2018), for tasks like object detection and semantic segmentation (Gu et al. 2022). The perception outcomes are fed into the action generator, where the robot formulates action strategies. There are different options for the action generator. Most studies adopt Recurrent Neural Networks (RNN) (Duan et al. 2022; Wijmans et al. 2019b), such as Long Short-Term Memory (LSTM) (Wijmans et al. 2019a; Wortsman et al. 2019; Wahid et al. 2021; Anderson et al. 2018) and Gated Recurrent Unit (GRU) (Khandelwal et al. 2022; Ye et al. 2021; Das et al. 2018), or transformers (Shao et al. 2023; Zhu et al. 2020; Fang et al. 2019). Recent works also propose to generate actions with diffusion-based methods (Ryu et al. 2024; Huang et al. 2023). Nevertheless, despite different types of action generators, they generally rely on the perception outputs from the visual encoders to make decisions on the actions. Thus, improving the efficiency of the visual encoding process is critical regardless what kind of actions generators are used in the vision-based mobile robotic systems. To meet the need for efficient visual encoding, we propose *Robye*, drawing inspiration from the human visual system. By incorporating motion-awareness, iconic memory, and a dichotomous approach, *Robye* enhances the efficiency and effectiveness of visual encoding in mobile robotic applications.

2.2 Visual encoders in robotics

In vision-based robotics, the development of visual encoders is becoming one of the pivotal studies at the intersection between computer vision and robotics (Gu et al. 2022). While it is commonly accepted that the visual encoders in robotics are original neural network models for computer vision tasks such as classification (Radosavovic et al. 2023; Khandelwal et al. 2022; Ryu et al. 2024; Huang et al. 2023) and semantic segmentation (Wijmans et al. 2019a; Li et al. 2022; Yen-Chen et al. 2020), recent works are focusing on escalating the training methods from supervised learning to self-supervised learning (Wang et al. 2024; Pari et al. 2021) such as contrastive learning (Khandelwal et al. 2022; Fung et al. 2023) and masked autoencoder (Radosavovic et al. 2023; Majumdar et al. 2024). While these studies keep the same neural architectures of the visual encoders as other works, they substitute the traditional supervised learning methods with self-supervised learning methods to train the parameters in the encoders in the training stage. Compared to the supervised trained encoders, the self-supervised trained encoders guide the robots to accomplish tasks with better performance (Khandelwal et al. 2022). Majumdar et al. (2024) and Nair et al. (2022) also explore on whether visual encoders trained by masked autoencoder are universal to different types of robotic tasks. It is important to note that, as we focus on improving the efficiency of the visual encoders in the inference stage, the changes in the training methods do not affect the design of the proposed framework.

2.3 Video deep neural networks

Video Deep Neural Networks (DNNs) are designed to analyze sequences of frames (Jain et al. 2019; Kim et al. 2022; Li et al. 2021; Rhee et al. 2022; Sun et al. 2022a), extending beyond single-image inputs by creating model architectures specifically tailored for video data (Awan and Shin 2021; Lee et al. 2021; Rhee et al. 2022; Zhuang et al. 2020). These models incorporate historical data from previous frames into the current frame analysis through lateral connections, bridging raw pixels (Kim et al. 2022; Petrovai and Nedevschi 2022; Wang et al. 2021; Woo et al. 2021; Xie et al. 2021; Ye et al. 2022; Liu et al. 2019, 2022; Yang et al. 2022; Du et al. 2020) or DNN features (Chen et al. 2022; Li et al. 2021; Rhee et al. 2022; Seong et al. 2021; Sun et al. 2022a; Zhou et al. 2022; Zhuang et al. 2023) between frames. However, such integration is resource-intensive (Kim et al. 2022; Sun et al. 2022b; Zhuang et al. 2023; Chen et al. 2022; Rhee et al. 2022; Zhou et al. 2022). Furthermore, most video DNNs deviate from the principles of single image-based DNNs due to their specialized neural architectures, losing compatibility (Awan and Shin 2021; Lee et al. 2021; Zhuang et al. 2020; Kim et al. 2022; Li et al. 2021; Rhee et al. 2022; Sun et al. 2022a). While some pixel-based video DNNs complement image-based models, they are constrained to the encoding type of 2D object detection (Liu et al. 2019, 2022; Yang et al. 2022; Du et al. 2020). These models also encounter considerable image processing challenges of identifying new

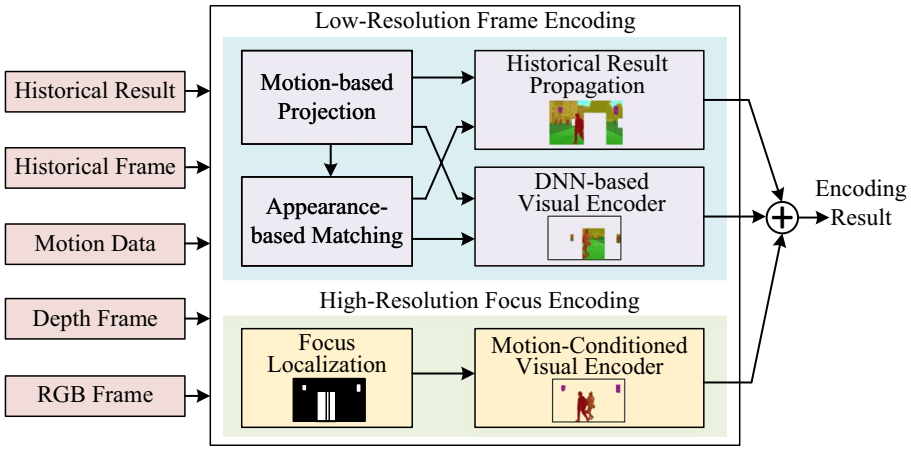


Fig. 3 *Robye*: a bionic visual encoding framework for mobile robotics

objects and tracking, leading to significant overhead (Liu et al. 2022; Yang et al. 2022).

Instead of relying solely on pixel information, we propose novel techniques that harness motion data from mobile robots' accelerometers and speed sensors, facilitating spatial geometric projection to correlate content, including both background and objects, across consecutive frames. These methods surpass pixel-based video DNNs in speed and sensitivity to new objects. Furthermore, we develop a comprehensive visual encoding framework tailored for mobile robots. This framework is designed to be compatible with single image-based DNNs and general to different encoding types. This framework incorporates a dual-resolution DNN processing approach alongside motion-conditioned encoding, significantly cutting down computational demands while enhancing the performance of tasks undertaken by mobile robotics. These designs not only streamline data processing but also boost the efficiency and effectiveness of robotic encoding and interaction within dynamic environments.

3 Robye overview

We propose *Robye*, a visual semantic encoding framework for mobile robotics, to enhance efficiency by mimicking human vision. As shown in Fig. 3, the framework has two workflows: low-resolution frame encoding (the 'purple' modules) and high-resolution focus encoding (the 'yellow' modules), with motion information and cached historical result (iconic memory) to boost efficiency. Specifically, in the frame encoding workflow, *Robye* aligns consecutive frames' background and objects using sensor-informed geometric projection in the motion-based projection module. The appearance-based matching module analyzes these aligned contents to localize new objects. *Robye* encodes the overlapping contents between consecutive frames by historical result propagation, and applies DNN-based visual encoding only on the non-overlapping content and new objects. In the focus encoding workflow, *Robye* localizes focusing areas based on application requirements and adaptively adjusts computational load on them conditioned by motion data. Given the encoding result from *Robye*, an action generator produces actions for mobile robotics.

4 Design of Robye

4.1 Low-resolution frame encoding

The low-resolution frame encoding in *Robye* emulates human peripheral vision and keeps full-frame understanding with low computational load. Specifically, it utilizes an iconic memory mechanism to transfer visual encoding result across frames through motion-based projection and appearance-based matching. By aligning the overlapping areas across frames, we propagate visual encoding result of the matched parts across frames and only process unmatched areas with DNN. Overall, by integrating motion-assisted correlation, iconic memory, and dual-level encoding, this workflow mirrors human vision efficiency to lower computational needs.

4.1.1 Motion-based projection

In the human visual system, motion plays a crucial role in shaping perception. Humans effectively track motion, both of themselves and objects in their surroundings, for predictive estimations of upcoming visual stimuli. This capability enables transferring information from past to future visual processing and reduces the computational load in visual encoding. In *Robye*, we mimic this process with a "motion-based projection" module, which incorporates motion into the visual encoding of robotic systems. To keep resource-efficient, this module identifies overlapping areas in both background and objects across frames, which are then

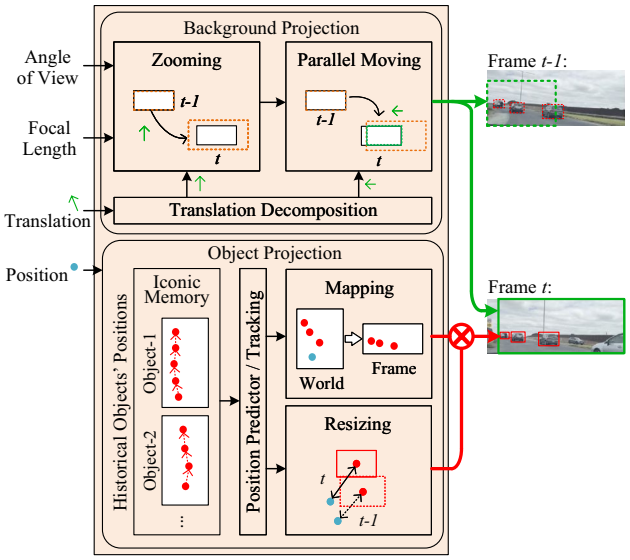


Fig. 4 Motion-based projection

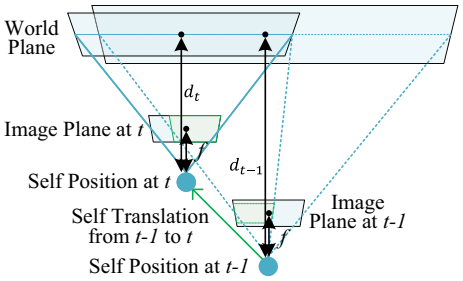


Fig. 5 Background projection

analyzed for appearance features in the "appearance-based matching" module.

As shown in Fig. 4, the motion-based projection is divided into the localization for the background and for objects. The background component, being static in the world coordinate system, changes position between frames due to the self movement. Conversely, object localization accounts for both the self movement and object dynamics within the environment. To localize object positions, we can adopt prediction or tracking techniques to estimate their temporal positions accurately. A plethora of options exists for these predictive and tracking approaches. In our implementation, we employ a hybrid way. Initially, an Object-oriented Fast Robust Binary (ORB) feature-based tracking method (Mur-Artal and Tardós 2014)[2] is used for newly appearing objects over three frames, with their world positions stored to the iconic memory. Subsequently, we employ a Kalman Filter (Welch et al. 1995) for predicting object positions, utilizing historical data from the iconic memory.

Background projection: Fig. 5 shows our use of the pinhole camera model to estimate the overlapping background area between consecutive frames. We record the self position at times t and $t-1$ as (x_t, y_t, z_t) and $(x_{t-1}, y_{t-1}, z_{t-1})$, with $\mathbf{s}$ representing the transition vector between these points. Assuming that the z-axis is perpendicular to the camera's image plane, $\mathbf{s}$ is split into $\mathbf{s}_z$ (along the z-axis, affecting zooming) and $\mathbf{s}_{xy}$ (on the xy-plane, causing parallel background shifts).

We calculate the effects of $\mathbf{s}_z$ and $\mathbf{s}_{xy}$ on the image plane using the camera-background distance d, camera view angles (α_x, α_y), and focal length f, obtained from intrinsic parameters. For estimating d, we consider the direct camera-wall distance in indoor settings, and average depth values from the background in outdoor scenes, using frame encoding results from the previous frame at $t-1$. We approximate d_t as $d_{t-1} \pm |\mathbf{s}_z|$, where "$-$" and "$+$" denote moving towards or away from the background, respectively. This approximation, as shown in Fig. 5, helps estimate the overlapping background area effectively.

With the orthogonal zooming vector $\mathbf{s}_z$, the four corners of the image plane at $t-1$, $\{(0,0), (0,l_y), (l_x,l_y), (l_x,0)\}$,[3] is zoomed to $\{(-\Delta z_x, -\Delta z_y), (-\Delta z_x, l_y + \Delta z_y), (l_x + \Delta z_x, l_y + \Delta z_y), (l_x + \Delta z_x, -\Delta z_y)\}$, where $\Delta z_x = \tan(\frac{\alpha_x}{2}) \cdot (\frac{d_{t-1}}{d_t} - 1) \cdot f$, $\Delta z_y = \tan(\frac{\alpha_y}{2}) \cdot (\frac{d_{t-1}}{d_t} - 1) \cdot f$, $l_x = 2f \cdot \tan(\frac{\alpha_x}{2})$, and $l_y = 2f \cdot \tan(\frac{\alpha_y}{2})$. With the parallel moving vector $\mathbf{s}_{xy}$, each 2D point (u, v) on the image plane at $t-1$ is shifted to $(u - \Delta s_x, v - \Delta s_y)$, where $\Delta s_x = \frac{f}{d_t} \cdot |\mathbf{s}_{xy}|_x$ and $\Delta s_y = \frac{f}{d_t} \cdot |\mathbf{s}_{xy}|_y$. The $|\cdot|_i$ represents the signed projection of a vector on the i-axis.

By combining the effects from the orthogonal zooming vector and the parallel moving vector, we obtain the mapping positions at t of the four image corners at $t-1$ as $\{(-\Delta z_x - \Delta s_x, -\Delta z_y - \Delta s_y), (-\Delta z_x - \Delta s_x, l_y + \Delta z_y - \Delta s_y), (l_x + \Delta z_x - \Delta s_x, l_y + \Delta z_y - \Delta s_y), (l_x + \Delta z_x - \Delta s_x, -\Delta z_y - \Delta s_y)\}$. With the four mapping corners of the image plane from $t-1$ to t, we obtain the intersection of union (IoU) between the two frames in frame t as:

$$I_t = \{\max(0, -\Delta z_x - \Delta s_x), \max(0, -\Delta z_y - \Delta s_y), \min(l_x, l_x + \Delta z_x - \Delta s_x), \min(l_y, l_y + \Delta z_y - \Delta s_y)\} \quad (1)$$

where the first two elements represent the xy-values of the bottom-left corner of the IoU in frame t and the last two elements represent the xy-values of the top-right corner. By

[2] The ORB feature-based tracking method is applied to 2D frames and we map it to the world position based on the robot's self positions.

[3] The original point is bottom-left corner and the indexing is clockwise.

mapping I_t to the image plane at $t-1$, we obtain the IoU between the two frames in frame $t-1$ as:

$$\begin{aligned} I_{t-1} = d_t/d_{t-1} \cdot \{ & \max(0, -\Delta z_x - \Delta s_x) + \Delta s_x, \\ & \max(0, -\Delta z_y - \Delta s_y) + \Delta s_y, \\ & \min(l_x, l_x + \Delta z_x - \Delta s_x) + \Delta s_x, \\ & \min(l_y, l_y + \Delta z_y - \Delta s_y) + \Delta s_y \} \end{aligned} \tag{2}$$

We visualize the procedure of background's projection between consecutive frames in Fig. 4.

Object projection: Objects in frames experience spatial changes due to the self movement and their own motion. We track or predict their world positions, denoted as (x_j^t, y_j^t, z_j^t) for object j at time t. Combined with the robot's position (x_t, y_t, z_t) and camera focal length f, we project the object's position in frame t as $(\frac{f}{(z_j^t - z_t)} \cdot x_j^t, \frac{f}{(z_j^t - z_t)} \cdot y_j^t)$, assuming the z-axis aligns with the camera's facing direction. We determine an object's area in frame t using the previous frame's visual encoding result, taking either the bounding box from object detection or the minimum box covering the semantic segmentation mask at $t-1$, mapping 3D bounding boxes to 2D planes for 3D detection. In this way, we obtain the size of the object j at $t-1$, denoted as (X_j^{t-1}, Y_j^{t-1}). We scale the size by the distance change from t to $t-1$, *i.e.*, $X_j^t = d_j^t/d_j^{t-1} \cdot X_j^{t-1}$ and $Y_j^t = d_j^t/d_j^{t-1} \cdot Y_j^{t-1}$, where d_j^t and d_j^{t-1} are the distance between the robot and the object j at t and $t-1$, respectively. We take the (X_j^t, Y_j^t) bounding box centering at $(\frac{f}{(z_j^t - z_t)} \cdot x_j^t, \frac{f}{(z_j^t - z_t)} \cdot y_j^t)$ as the memonic part of object j in frame t. We visualize the procedure of objects' projection between consecutive frames in Fig. 4.

It is important to note that the motion-based projection in *Robye* are not designed for exact overlapping region identification between frames. Instead, we aim to cost-effectively identify potential overlapping areas considering self-motion and object motion patterns. These candidate overlapping parts are then further examined by the appearance-based matching module to determine if they require DNN-based visual encoding.

4.1.2 Appearance-based matching

The appearance-based matching module receives as input the potential overlapping parts identified by the motion-based projection module. As shown in Fig. 6, we design individual comparative analysis for background and object matching, respectively.

Background matching: For background matching, we scale and align the background part from the previous frame, $\mathbf{X}_B^{t-1}$, with the motion-based overlapping area $\mathbf{X}_B^t$ in the current frame, as depicted in Fig. 6's "Grid Background Matching". We divide both $\mathbf{X}_B^{t-1}$ and $\mathbf{X}_B^t$ into an $N \times M$ grid,

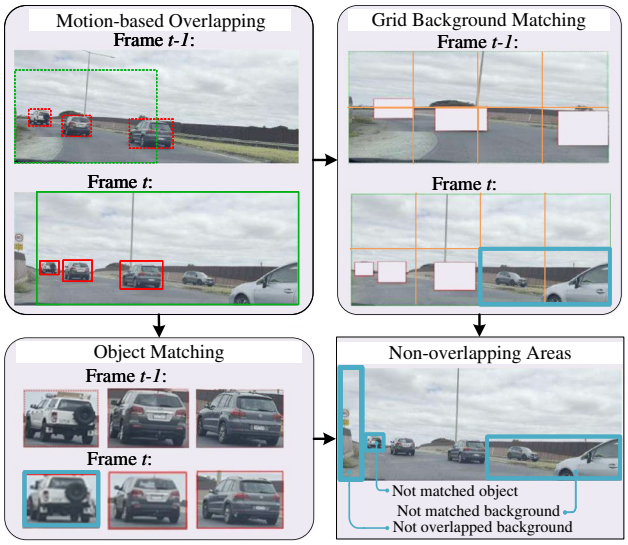


Fig. 6 Appearance-based matching

calculating element-wise differences between corresponding grid cells. A predefined threshold on this difference determines a match; for example, non-matching cells are highlighted in blue in Fig. 6. When computing average differences, (potential) object parts are nullified by setting their differences to zero.

Object matching: For object matching, we use Singular Value Decomposition (SVD) for assessing object correspondence between time steps $t-1$ and t. Our empirical evaluations suggest that SVD yields superior performance in measuring object similarity compared to element-wise difference with trivial computational overhead. We represent object j at time $t-1$ as $\mathbf{X}_j^{t-1}$ and at time t as $\mathbf{X}_j^t$, decomposing them into $Ut-1\Sigma t-1V_{t-1}^T$ and $U_t\Sigma_t V_t^T$. The singular values in Σ_* reflect the importance of basis vectors in representing $\mathbf{X}_j^{t-1}$ and $\mathbf{X}_j^t$. We calculate the Frobenius norm[4] of the difference between $\Sigma t-1$ and $\Sigma t-1$, with a smaller value indicating higher similarity. We set a threshold to decide whether the objects of $t-1$ and t match.

4.1.3 Historical result propagation

For the matched regions within the background and objects, we derive their visual encoding result by employing a process of translation and scaling, which is executed in accordance with the motion-based projection module. This procedure involves the transformation of the relevant result from the previous frame to align with the current frame's context, which enables the direct generation of visual encoding result.

[4] The Frobenius norm is defined as the square root of the sum of squares of all singular values within Σ_*.

4.1.4 DNN-based visual encoder

Non-overlapping areas in *Robye* fall into three types: (1) areas not projected by the motion-based projection module, (2) background parts that do not match in the appearance-based matching module, and (3) object parts that also remain unmatched in the appearance-based matching module. Figure 6 visualizes examples of non-overlapping areas in blue. Notably, new objects appearing in each frame are effectively captured as non-overlapping areas through our grid background matching method. For these non-overlapping areas, *Robye* utilizes a DNN to obtain visual encoding results. As a general framework for robotic visual encoding, *Robye* allows for the integration of various DNN within its DNN-based visual encoder.

Our low-resolution frame encoding workflow has two distinct sources for the final results: (1) Overlapping part results from the historical result propagation module, and (2) Non-overlapping part results via the DNN-based visual encoder module. The complete encoding result for a frame comes from merging these two. Since there is no overlap between the parts, their combination is based on their spatial positions.

4.2 High-resolution focus encoding

The focus visual encoding workflow in *Robye* involves two steps. First, it uses a cost-effective model to localize focusing regions with bounding boxes. Second, the workflow adaptively adjusts computational load for DNN-based visual encoding based on motion change magnitude, to reflect how position changes affect feature variation over time. As motion changes increase, the computational cost rises, while the influence of historical features decreases.

4.2.1 Focus localization

In mobile robotics, such as autonomous driving, understanding the spatial relationships between the robot, nearby objects, and the background is key to identifying focusing regions. Objects closer to an autonomous vehicle are more crucial for immediate decisions, while distant objects can still be significant in vision and language navigation. Consequently, focus localization is split into two types: distance-relevant, focusing on proximate objects, and distance-irrelevant. Figure 7 demonstrates this, categorizing vehicles and pedestrians as distance-relevant and traffic signals and signs as distance-irrelevant.

In mobile robotics, human experts play an important role in creating ground-truth annotations for training datasets, especially for tasks resembling human activities. We involve

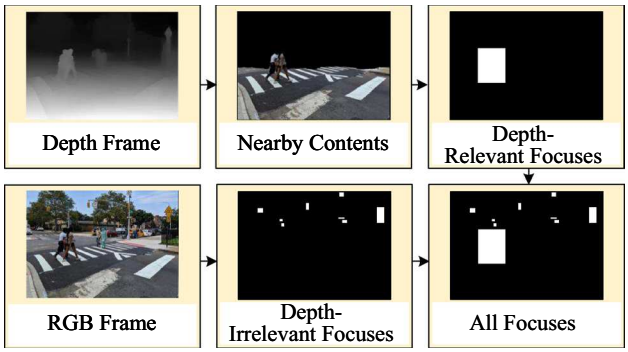


Fig. 7 A focus localization example

human experts to define ground-truth focusing regions for training our focus localization model. However, unlike labor-intensive dataset labeling, we only require experts to identify object categories for focusing regions and set distance thresholds. Using an object detection model, we automate the annotation of ground-truth bounding boxes in the dataset, by omitting semantic class labels and retaining only the coordinates. This process yields two sets of ground-truth bounding boxes, one for each localization type. These bounding boxes, termed "binary masks," lack semantic labels, dividing image content into focusing regions and the rest.

To design the focusing region localization model, we modify the output layer of an object detection model, by removing neurons for semantic labels and keeping those for bounding box coordinates. The model is trained using a bounding box regression loss function, expressed as $L = (b_1 - b_{1,gt})^2 + (b_2 - b_{2,gt})^2 + (b_3 - b_{3,gt})^2 + (b_4 - b_{4,gt})^2$, where b_1, b_2, b_3, b_4 denote the four coordinate values defining the bounding box, and $b_{*,gt}$ represents their corresponding ground truth counterparts. The removal of semantic recognition allows for model compression with minimal IoU loss. For example, we compress the Tiny-YOLO model by about 63.7% in operations, with only a minor IoU reduction of less than 7.5%. This demonstrates the efficiency of our binary-masking focus localization model.

4.2.2 Motion-conditioned visual encoder

Mirroring the human visual system's foveation acuity, we feed high-resolution focusing regions into our motion-conditioned visual encoder module. Using motion data from the motion-based projection module, we integrate historical features of focusing regions into the encoding process. This integration adapts to motion change magnitudes in the environment, as shown in Fig. 8. If a focusing region includes an object from the previous frame, we use its historical features. These features, termed "historical features" and denoted by $\mathbf{X}_p$, are

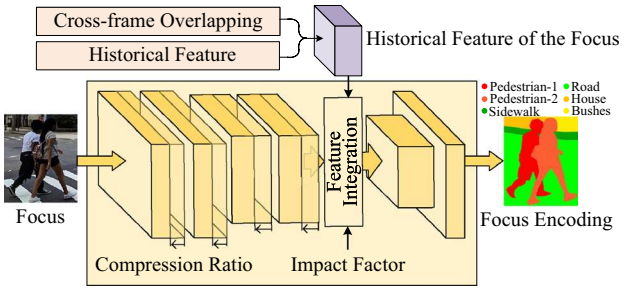


Fig. 8 Motion-conditioned visual encoder

directly accessible based on the motion-based projection and appearance-based matching modules.

In *Robye*, we offer the flexibility to employ any DNN model within the motion-conditioned visual encoder module. To curtail computational overhead, we employ compression techniques (He et al. 2017a) at the first K layers of the DNN model, where K is a hyper-parameter. The degree of compression is governed by the compression ratio c, evenly ranging from $c_0 (> 0)$ to 1 with $\mathbf{C}$ levels. $\mathbf{C}$ is a hyper-parameter that defines the number of optional compression ratios. For example, the adjustment of the number of channels in each convolutional layer of the K layers of the DNN model follows the formula $\lceil c \cdot D \rceil$, where D is the original number of channels in the layer. The adaptation of the compression ratio c is contingent upon the magnitude of observed motion changes, referred to as the "motion-conditioned compression ratio". Specifically, we select the compression ratio based on a motion index $\mathcal{M}$ defined as:

$$\mathcal{M} = \left\| \frac{|\mathbf{m}^t|_x}{|\mathbf{m}^t|_y} - \frac{|\mathbf{m}^{t-1}|_x}{|\mathbf{m}^{t-1}|_y} \right\| \cdot \| |\mathbf{m}^t| - |\mathbf{m}^{t-1}| \| \tag{3}$$

where $\mathbf{m}^t$ is the vector from the self-position (x_t, y_t, z_t) to the focusing object's position (x_j^t, y_j^t, z_j^t) at time t, $\mathbf{m}^{t-1}$ is the vector at time $t-1$, $|\cdot|_i$ represents the signed projection of a vector on the i-axis, $|\cdot|$ represents the length of the vector, and $\|\cdot\|$ represents the absolute value. The term $\left\| \frac{|\mathbf{m}^t|_x}{|\mathbf{m}^t|_y} - \frac{|\mathbf{m}^{t-1}|_x}{|\mathbf{m}^{t-1}|_y} \right\|$ indicates alterations in the view angle of the self-robot in relation to the focusing object from $t-1$ to t, while the term $\| |\mathbf{m}^t| - |\mathbf{m}^{t-1}| \|$ characterizes changes in the viewing distance between the self-robot and the focusing object from $t-1$ to t. The compression ratio is one of the $\mathbf{C}$ compression levels from c_0 to 1 that is nearest to $c_0 + \max\{\gamma_1 \cdot \mathcal{M}, 1 - c_0\}$:

$$c = \arg\min_d \{c_0 + \max\{\gamma_1 \cdot \mathcal{M}, 1 - c_0\} - c_d\}_{d \in [0, \mathbf{C}]}, \tag{4}$$

$$c_d = c_0 + \frac{d}{\mathbf{C}} \cdot (1 - c_0). \tag{5}$$

where γ_1 is a hyper-parameter to control the effect of motion on the compression ratio c. Our empirical observations show that both terms in $\mathcal{M}$ lead to variations in feature representations. Thus, they generate an augmented computational cost (higher c_0), in cases where these changes are significant in their magnitudes.

In the DNN model, we insert an feature integration layer between its original K-th layer and $(K+1)$-th layer, as shown in Fig. 8. The output of the integration layer $\mathbf{X}_I$ equals to:

$$\mathbf{X}_I = \mathbf{X}_K + \gamma_2 \cdot \frac{1}{\mathcal{M}} \mathcal{F}(\mathbf{X}_p, \mathbf{X}_K) \tag{6}$$

where γ_2 is the impact factor to control the effect of motion on the integration of the historical feature $\mathbf{X}_p$, and $\mathbf{X}_K$ denotes the output from the first K layers in the DNN model. It is noteworthy that $\mathbf{X}_K$ is affected by the motion-conditioned compression ratio c in Eq. (4). Although a direct integration of $\mathbf{X}_p$ into $\mathbf{X}_K$ by setting $\mathcal{F}(\mathbf{X}_p, \mathbf{X}_K) = \mathbf{X}_p$ is feasible, our empirical observations reveal its suboptimal performance. Instead, we employ a cross-attention layer to construct $\mathcal{F}(\mathbf{X}_p, \mathbf{X}_K)$, a strategy that demonstrates enhanced performance gains without incurring substantial computational overhead. It is worth emphasizing that we have aligned the dimensions between $\mathbf{X}_p$ and $\mathbf{X}_K$ by bicubic interpolation. In $\mathcal{F}(\mathbf{X}_p, \mathbf{X}_K)$, $\mathbf{X}_K$ is the "Query" for information retrieval, and $\mathbf{X}_p$ acts as both "Key" and "Value" for reference and aggregation. The output merges $\mathbf{X}_p$ and $\mathbf{X}_K$ features with contextually relevant elements from $\mathbf{X}_K$.

We train the cross-attention layer parameters and fine-tune the compressed parameters of the DNN's first K layers end-to-end with the same loss function as the original visual encoding task. Parameters beyond the K layers are frozen, with loss gradients computed only for the first K layers and cross-attention layer during backpropagation. We also offer parameter sharing across different compression levels (Fang et al. 2018). Training data is divided into $\mathbf{C}$ groups by motion-conditioned compression ratios, and the first K layers' parameters are incrementally trained from c_0 to 1 across $\mathbf{C}$ steps. Each step freezes parameters from previous iterations and fine-tunes current step parameters, to keep the total parameter count in the first K layers unchanged. Training data is prepared from consecutive frames of the original dataset, with object detection localizing focusing regions. Historical feature extraction from these regions in low-resolution preceding frames is derived directly from features in the low-resolution frame encoding workflow.

5 Implementation

Applications: We evaluate *Robye* on three popular mobile robotic applications: *(1) Autonomous Driving:* We test on CARLA public leaderboard (https://leaderboard.carla.org/). We also build the autonomous driving on a real vehicle. The inputs of visual encoding include three cameras facing front, left, and right and a LiDAR sensor. The original RGB resolution is 800×600. *(2) Vision-and-Language Navigation (VLN):* We test on R2R dataset (Anderson et al. 2018). The original RGB resolution is 640×480. *(3) Path Planning of Drone:* We test on LANI (Misra et al. 2018). The original RGB resolution is 256×144. We also build a drone with path planning for real-world test.

Types of visual encoding: We include six types of visual encoding: *(1) 2D Object Detection:* We evaluate on YOLO-v5, EfficientDet (Tan et al. 2020), and Faster-RCNN (Girshick 2015). *(2) 3D Object Detection:* We evaluate on FrustumPointNet (Qi et al. 2018), Mono3D (Chen et al. 2016), and MLCVNet (Xie et al. 2020). *(3) 2D Instance Segmentation:* We evaluate on Mask-RCNN (He et al. 2017b), SOLO-v2 (Wang et al. 2020), and YOLACT (Bolya et al. 2019). *(4) 3D Instance Segmentation:* We evaluate on Mask-RCNN 3D (He et al. 2017b), PointNet-Seg (Qi et al. 2017), and 3D-BoNet (Yang et al. 2019). *(5) Semantic Segmentation:* We evaluate on FCN (Long et al. 2015), SegNet (Badrinarayanan et al. 2017), and DeepLab-v3 (Chen et al. 2019). *(6) Backbone*[5]: We evaluate on ResNet-101 (He et al. 2016), ResNet-50 (He et al. 2016), and VGG (Simonyan and Zisserman 2014). *It is important to note that Robye applies to both 2D image data and 3D sensing data (e.g., LiDAR data).* In the motion-based projection module, the overlapping parts of 3D sensing data in background and objects can be directly obtained based on self translation. In the high-resolution focus encoding workflow, we map the focusing regions on the 2D image from the focus localization module to the 3D sensing data based on the camera's angle of view and focal length. We accelerate all the models with TensorRT (NVIDIA TensorRT 2024).

Action generation model: The inputs into the action generation model include both frame and focus encoding results. LSTM directly takes the output vector(s) from the encoding workflows as the input. The LSTM is with two hidden layers and each layer has 512 hidden units. We have 32 action embedding. For VLN and drone path planning, we have 256 word embedding. We train the action generation models following their typical training scheme (Anderson et al. 2018).

[5] For memonic part localization, we add a lightweight 2D object localization head with three additional layers upon the output from the backbone to generate proposals of objects' locations without semantic recognition.

Embedded edge/mobile devices: We evaluate on four embedded GPU devices: NVIDIA Jetson Nano, Jetson TX2, Jetson Xavier, and Jetson Orin. We also evaluate on two mobile devices: Samsung Galaxy S22 (https://www.samsung.com/us/smartphones/galaxy-s22/) and Google Pixel 7 (https://store.google.com/product/pixel_7?hl=en-US).

Edge-assist setup: We evaluate with two options on the device: NVIDIA Jetson Nano and Google Pixel 7. The edge server is equipped with NVIDIA 1080 Ti. We evaluate with two options on the wireless network: WiFi 5 and 5 G. In this setup, we offload the DNN-based visual encoding modules to the edge server and keep the other modules on the device.

Metrics: For object detection and instance segmentation, the metric is mAP. For semantic segmentation, the metric is mIoU. For autonomous driving, the metric is driving score, which is the product of route completion ratio and infraction score (Shao et al. 2023). For vision-and-language navigation, the metric is success rate weighted by path length (SPL) (Anderson et al. 2018). For drone navigation, the metric is success rate (Blukis et al. 2018).

Low-resolution settings: For autonomous driving, the resolution is scaled down to 240×240. For VLN, the resolution is scaled down to 224×224. For drone navigation, the resolution is scaled down to 72×72.

Settings in appearance-based matching: We set $N = M = 4$ for the grid background matching. We set the difference threshold as 17 for background matching and as 2.4 for object matching.

Settings in motion-conditioned visual encoding: For autonomous driving, the focusing regions are cropped from the 520×520 frame resolution. For VLN, the focusing regions are cropped from the 400×400 frame resolution. For drone navigation, the focusing regions are cropped from the 144×144 frame resolution. For each model, we compress the first 70% layers into 5 compression levels and $c_0 = 0.3$. We set $\gamma_1 = 0.4$ and $\gamma_2 = 0.5$.

Baselines: We selected two leading works in efficient object detection to serve as baselines against our low-resolution frame encoding workflow inside *Robye*. The first baseline, FlexPatch (Yang et al. 2022), focuses on on-device object detection. However, unlike *Robye*, it lacks integration of physical motion awareness and relies on a combination of image processing techniques and a decision tree classifier for object localization across frames. While FlexPatch employs a cropping strategy to reduce detection latency, it incurs large overhead for object localization and its edge-intensity-based object discovery method falls short in robustness. The second baseline, AdaMask (Liu et al. 2022), focuses on object detection offloading but does not incorporate new object discovery, opting instead for periodic full-frame offloading, which results in diminished accuracy. Additionally, lacking geometric projection informed by physical motion, AdaMask arbitrarily increases the crop size based on

tracking distance, leading to further latency increases. For autonomous driving and VLN, we set the input resolution to 320×320 for all baselines. For drone navigation scenarios, the resolution is 100×100.

6 Evaluation

6.1 Performance on autonomous driving

The performance of autonomous driving on the CARLA public leaderboard (https://leaderboard.carla.org/) is shown in Fig. 9 for 2D visual encoding models and Fig. 10 for 3D models. Note that in Fig. 10i–k, the visual encoder employs a ResNet-18 (Shao et al. 2023) to extract LiDAR Bird's Eye View (BEV) features in addition to 2D visual features. Thus, we categorize it under 3D visual encoding models.

2D Visual encoding: As shown in Fig. 9, we measure the average latency per action decision, frame accuracy, and driving score with three types of 2D visual encoding on four mobile and embedded devices. For 2D object detection, we present results using YOLO-v5, EfficientDet, and Faster-RCNN in Fig. 9a–g. Figure 9h–n show the performance with 2D instance segmentation models including Mask-RCNN, SOLO-v2, and YOLACT. We evaluate the performance with semantic segmentation, utilizing models of FCN, SegNet, and DeepLab, as shown in Fig. 9o–u. It is important to note that for each action, we process three RGB frames from the front, left, and right camera angles through the visual encoding model.

Robye enhances driving scores by +3% to +11% and achieves 1.5 to 2.6× faster latency than the model-only baseline, due to its motion-aware cross-frame mapping, dualistic frame and focus encoding workflows, and motion-conditioned focus encoding. The reasons for *Robye* 's superior performance are:

(1) Motion-based projection: Robye leverages motion-based cross-frame content correlation for efficient spatial projection, processing only non-overlapping areas with DNN. As the non-overlapping parts are often a small portion of a frame, the computing latency of the non-overlapping encoding is significantly lower than processing the whole frame, as shown in Fig. 9.

(2) Dual-level encoding: Different from the single-level 320×320 resolution encoding of the baseline model, *Robye* adopts a two-tier approach: low-resolution (240×240) holistic and high-resolution focus encoding (crops from 520×520). While the overall frame accuracy of *Robye* may be lower than the baseline, its focus accuracy excels due to higher resolution (Fig. 9g, n, u). This dual encoding finally contributes to *Robye* 's improved driving scores.

(3) Motion-conditioned focus encoding: The focusing regions in *Robye*, despite being high-resolution, are small crops from the entire frame, making the input size relatively small. For example, a 75×75 focusing region from a 520×520 frame is just 5.5% of a 320×320 frame's size. Moreover, *Robye* 's DNN computational cost in the focus encoding is dynamically adjusted based on motion change magnitude, which further enhances efficiency.

Robye surpasses the FlexPatch baseline with a +16% to +25% improvement in driving scores and a 1.3 to 2.2× speedup in latency. This is achieved through its *Dual-Level Encoding* and *Motion-Conditioned Focus Encoding* as analyzed above. Additionally, our *Motion-based Projection* also contributes to the higher performance of *Robye* over FlexPatch. Specifically, as FlexPatch uses image processing methods including edge intensity, optical flow, and a decision tree classifier trained on image analysis, together for tracking objects across frames, it suffers from high latency overhead and limited new object detection capabilities. It also lacks application to semantic segmentation (Fig. 9o–u) as FlexPatch is designed for object detection in camera surveillance and intrinsically ignores background.

In contrast, *Robye* employs geometric relationships and motion data for cost-effective cross-frame content correlation, applying image-based comparisons only to geometrically correlated areas. This method detects new objects and backgrounds using comparisons of element-wise differences and SVD. *Robye* 's use of a 240×240 resolution for frame encoding, compared to FlexPatch's 320×320, partially contributes to its latency advantage. However, FlexPatch's limitations in new object discovery and tracking accuracy result in lower overall performance, even with the higher input resolution. When using the same resolution, *Robye* still outperforms FlexPatch with a 1.2 to 1.7× speedup and an even greater driving score improvement of +18% to +29%.

3D visual encoding: As shown in Fig. 10, we measure the average latency per action decision, encoding accuracy, and driving score with three types of 3D visual encoding on two mobile/embedded devices and two edge-assisted setups. For 3D object detection, we present results using FrustumPointNet, Mono3D, and MLCVNet in Fig. 10a–d. Figure 10e–h show the performance with 3D instance segmentation models including Mask-RCNN 3D, PointNet-Seg, and 3D-BoNet. We evaluate the performance with 2D/3D backbone, utilizing models of ResNet-101, ResNet-50, and VGG, as shown in Fig. 10i–l. These 3D encoding models use inputs in formats like RGB, depth, or point cloud. For RGB and/or depth based models, we have inputs from three cameras, while we have a single environmental point cloud for point cloud based models.

On mobile/embedded devices, with *Motion-based Cross-Frame Projection*, *Dual-Level Encoding*, and

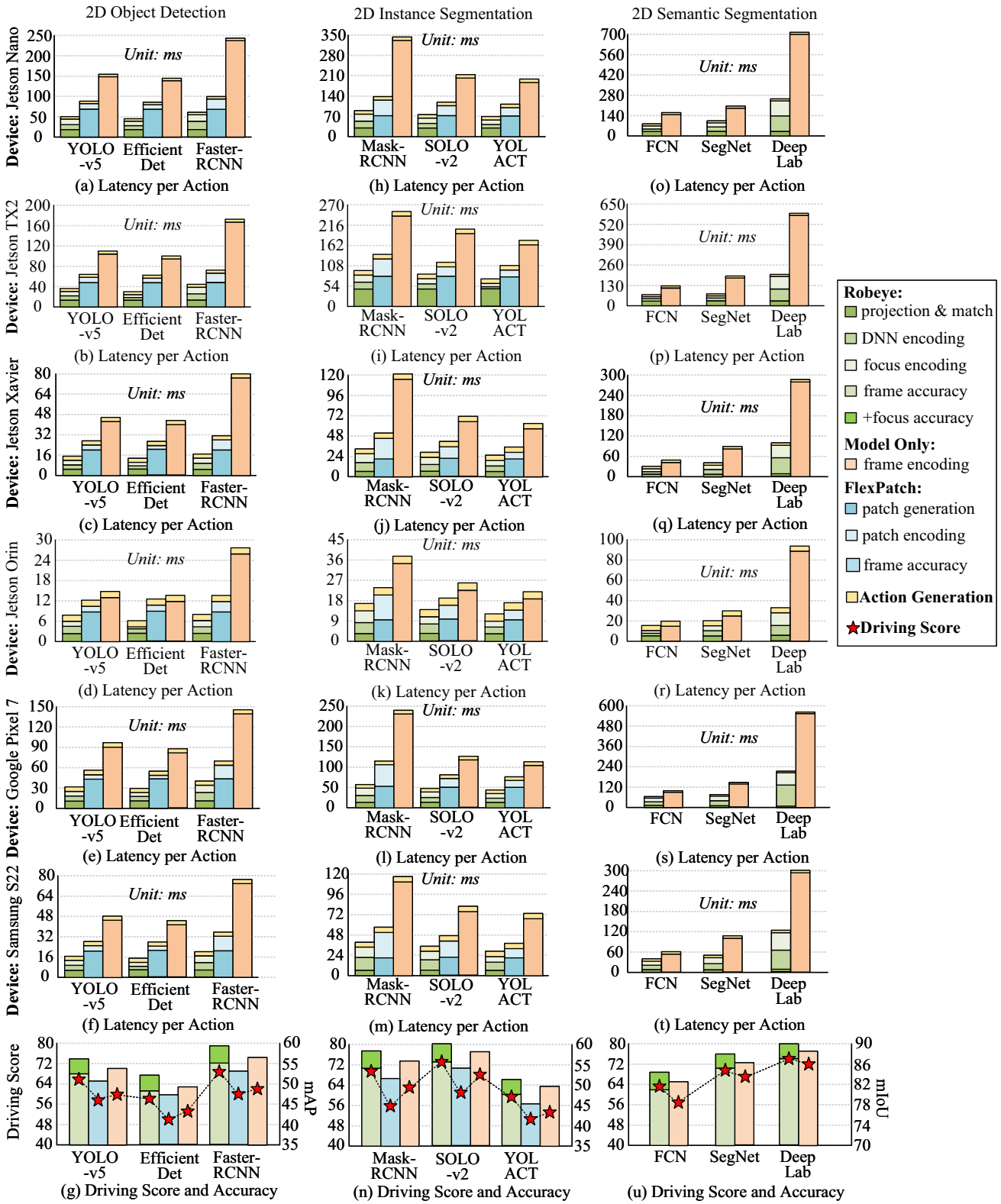


Fig. 9 Performance of autonomous driving with 2D visual encoding

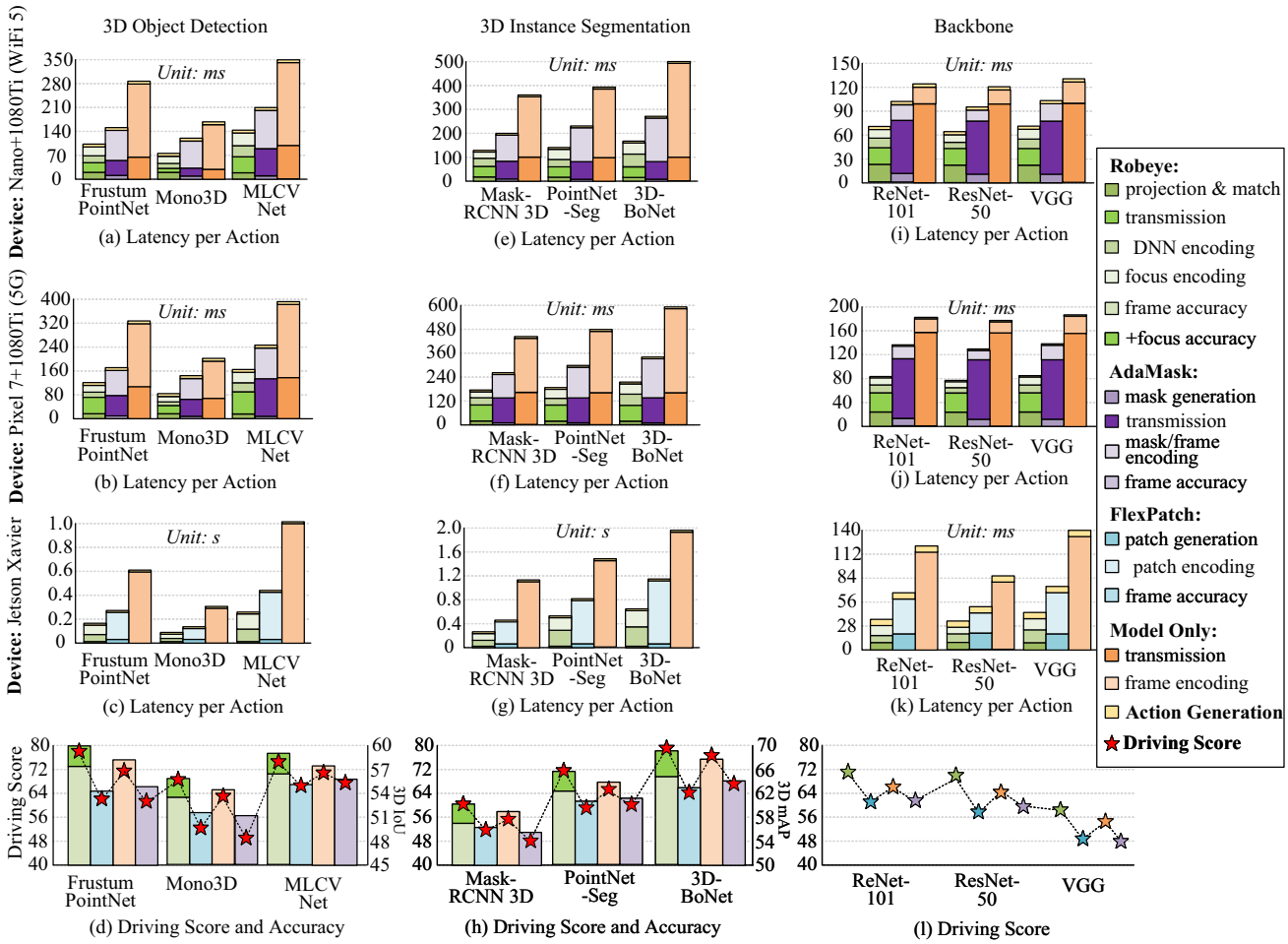


Fig. 10 Performance of autonomous driving with 3D visual encoding

Motion-Conditioned Focus Encoding, *Robye* achieves notable performance enhancements. Compared to the model-only baseline, *Robye* improves driving scores by +3% to +10% and reduces latency by 2 to 3.3×. Compared to the FlexPatch baseline, it boosts driving scores by +12% to +26% and achieves 1.3 to 1.9× faster latency. The reasons for *Robye* 's superior performance are the same as those in 2D visual encoding.

In edge-assisted setups, with *Motion-based Cross-Frame Projection*, *Dual-Level Encoding*, and *Motion-Conditioned Focus Encoding*, *Robye* outperforms baselines. Specifically, it achieves 1.6–2.8× faster latency compared to the model-only baseline and improves driving scores by +17% to +36% while also achieving 1.4–1.6× faster latency than the AdaMask baseline. AdaMask's limitations include: (1) Lack of New Object Localization Capability: AdaMask periodically offloads full-frame data for encoding, increasing latency. This approach often misses new objects, impacting accuracy due to its periodic nature. (2) Cropping Area Expansion: AdaMask expands cropping areas based on the position difference from the last detected result on the image plane, which often including more background content. These larger, unnecessary cropped areas further increase transmission and computing latency. These drawbacks explain why *Robye*, with its motion-based cross-Frame projection, surpasses AdaMask in performance on edge-assisted setups.

6.2 Performance on VLN and drone navigation

In our evaluation of *Robye* on Vision-and-Language Navigation (VLN) and drone navigation (Figs. 11 and 12), we test 3D object detection and semantic segmentation models on a Jetson Xavier embedded GPU and an edge-assisted setup with Google Pixel 7 and 1080 Ti. For 3D object detection, *Robye* surpasses the model-only baseline with a 1.4–2.7× latency speedup and a +3% to +6% performance increase. It improves latency by 1.5 to 1.9× speedup and performance by +6% to +9% over the AdaMask baseline, and shows a 1.7 to 2.2× latency speedup and +7% to +8% performance enhancement compared to the FlexPatch baseline. With

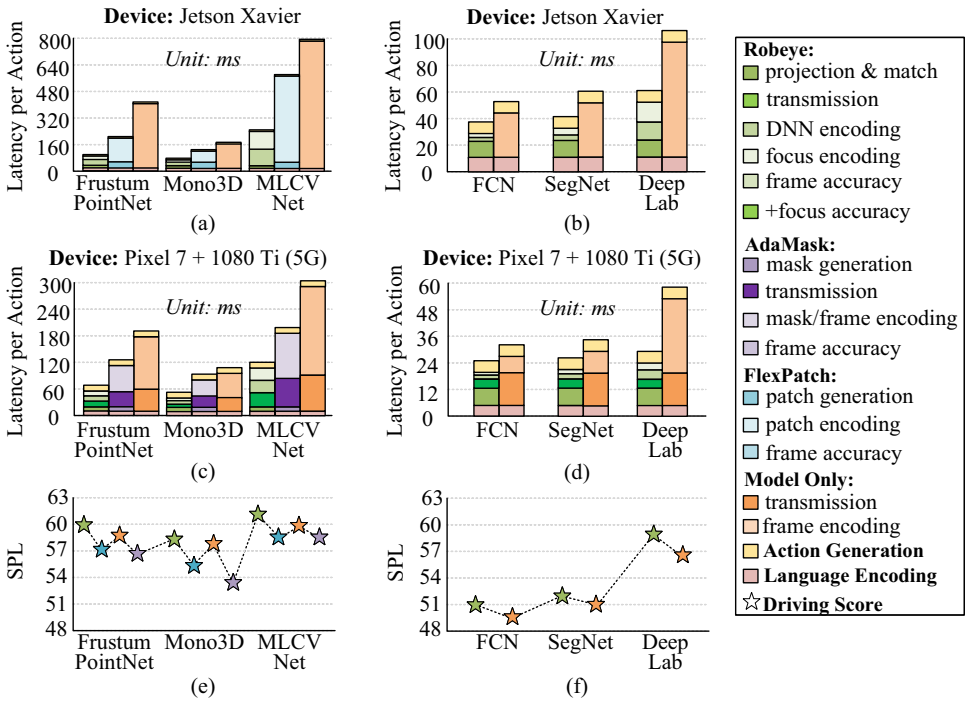


Fig. 11 Performance of vision-and-language navigation

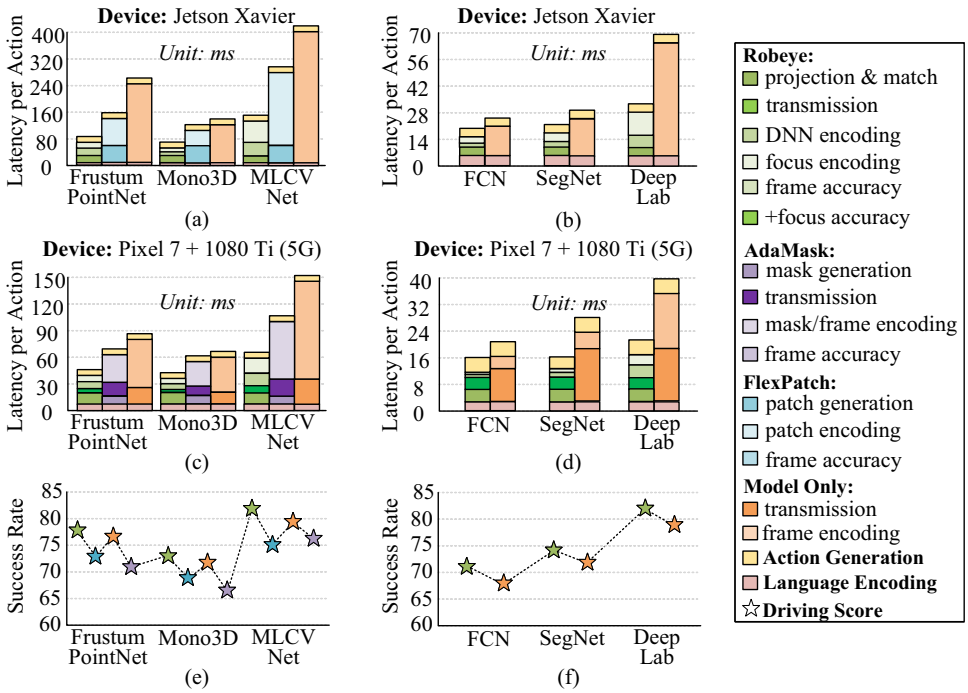


Fig. 12 Performance of drone navigation

semantic segmentation models, *Robye* outperforms the model-only baseline with a 1.3–2× latency speedup and a +4% to +7% performance increase. Besides the reasons discussed in Sect. 6.1, FlexPatch's reliance on edge intensity for new object discovery leads to excessive patch generation in indoor environments, further resulting in speed and performance losses in VLN scenarios.

6.3 Performance on real robotic platforms

To evaluate the performance of *Robye*, we test them on two real hardware platforms: (1) In the car setup (shown in Fig. 13a, b), we utilize three Intel Realsense L515 (https://www.intelrealsense.com/lidar-camera-l515/) as the capturing device, all these cameras connect to the Jetson AGX

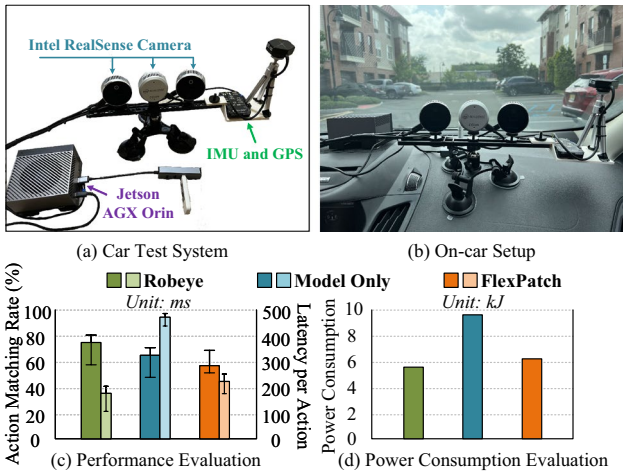


Fig. 13 Real car test on autonomous driving

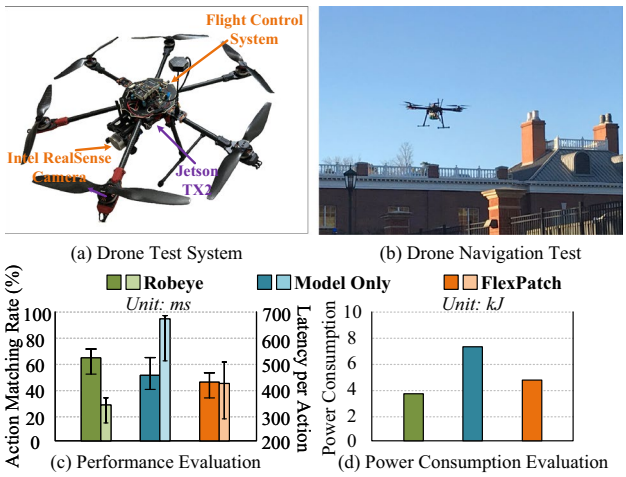


Fig. 14 Real drone test for navigation

Orin (https://www.nvidia.com/en-il/autonomous-machines/embedded-systems/jetson-orin/) through USB 3.1 cables. We take 3D-BoNet as the visual encoder. The Inertial Measurement Unit (IMU), and GPS sensor are also attached to this system. During the road test, we collect the actions from the human driver and *Robye* for action matching rate calculation. (2) In the drone setup (shown in Fig. 14a, b), we design and build a Hexacopter as the testing platform for drone navigation. The Hexacopter is equipped with six 4006 motors and a Pixhawk (https://pixhawk.org/) flight controller. The capturing device Intel Realsense L515 is connected to an on-board Jetson TX2. We take Mono3D as the visual encoder. The Jetson TX2 communicates with Pixhawk flight controller through I2C protocol. We evaluate *Robye* and the baselines on performance in Figs. 13c, d and 14c, d. Overall, *Robye* outperforms the baselines by 9% to 18% higher matching rate, 2.2 to 2.7× latency speedup, and 36% to 47% power consumption reduction (20 min).

6.4 Ablation study

We investigate the individual contributions of key modules within *Robye* to its overall performance enhancement. We selectively remove each module and observe the resultant impact on performance, as detailed in Table 1, on the autonomous driving application using Mono3D (Chen et al. 2016). Based on the observations, we analyze the contributions of each module as follows:

Motion-based projection: The module identifies overlapping areas between consecutive frames based on the robot's movement. Instead of running a DNN model to encode these areas, *Robye* directly propagates the encoding results from the previous frame. By reducing the DNN model's input size, we lower the computation required. On average, the number of operations in the DNN encoder decreases from 19.2 to 13.7 GFLOPs per frame in the low-resolution encoding workflow with the motion-based projection module. This reduction in computation translates to a decrease in latency per action from 113.9 to 85 ms. Furthermore, this reduction in computation has only a minimal negative impact on accuracy (3D IoU) and driving score, as we approximate the encoding results of the overlapping areas rather than ignoring them.

Appearance-base matching: The module further refines the overlapping areas identified by the motion-based projection module using image comparison. It is designed to detect areas with new content within the overlapping regions, *i.e.,* false positives. Without the appearance-based matching module, the overlapping areas are larger, including these false positives, which results in reduced DNN computation. On average, the number of operations in the DNN encoder increases from 12.8 to 13.7 GFLOPs per frame in the low-resolution encoding workflow when using the appearance-based matching module. However, identifying new content is crucial for maintaining a high driving score as ignoring objects can lead to infractions and route completion failures. Based on our observation, without the appearance-based matching module, the driving score drops from 69 to 48 due to missed obstacles and objects. Thus, the appearance-based matching module plays an important role in improving the driving score with trivial increase of computation and latency.

Motion-conditioned encoder: The module adaptively adjusts the DNN computation for processing the focusing regions based on position changes between consecutive frames. By reducing computation for small movements of the robot, the average number of operations in the DNN encoder decreases from 21.3 to 15.1 GFLOPs per frame

Table 1 Ablation study: *Robye* w/ a module vs *Robye* w/o a module (GLOPs and 3D IoU are compared in low-resolution and high-resolution encoding workflows)

Robye	GFLOPs (low-r, high-r)	IoU (low-r, high-r)	Latency (ms)	Driving Score
Original *Robye*	13.7, 15.1	53, 56	85	69
w/o Motion-based projection	19.2, 15.1	53.5, 56	113.9	69.5
w/o Appearance-based matching	12.8, 15.1	47, 52.8	82.7	48.0
w/o Motion-conditioned encoder	13.7, 21.3	53, 56.4	109.6	71.1

in the high-resolution encoding workflow when using the motion-conditioned encoder. This reduction in computation leads to a decrease in latency per action from 109.6 to 85 ms. The motion-conditioned encoder shows a slight decrease in accuracy (3D IoU) from 56.4 to 56 and a drop in driving score from 71.1 to 69.

7 Discussion

Robye **for different robotic applications:** The development of a universally applicable visual foundation DNN model for robotic applications remains a challenging task (Xu et al. 2024; Majumdar et al. 2024). The performance improvement of *Robye* primarily comes from two designs: (1) reducing computation on overlapping areas across frames, and (2) high-resolution encoding on focus regions only. In low-speed applications such as indoor and downtown navigation, overlapping areas between frames are large, which leads to more computation reduction. In contrast, high-speed applications such as highway driving have smaller overlapping areas, which leads to less computation reduction. Similarly, applications with more focusing regions consume more computing resources. Thus, to keep fast response for different applications, we need to adaptively adjust the available computing resources for them. For example, by profiling the relationship between the DNN-based encoder's computing latency and input size offline, we can prepare more computing resources to high-speed applications and those with more focus regions.

Robye **for different DNN architectures:** The design of *Robye* is orthogonal to DNN-based embodied AI systems and is adaptable to visual encoders of different DNN architectures, including CNNs and Vision Transformers (ViT). Using a pre-trained DNN-based visual encoder (Radosavovic et al. 2023; Khandelwal et al. 2022; Ryu et al. 2024; Huang et al. 2023; Wijmans et al. 2019a; Li et al. 2022; Yen-Chen et al. 2020), we directly take it as the DNN-based visual encoder in the low-resolution frame encoding workflow. We prepare a dynamic version of the encoder with multiple compression levels following (He et al. 2017a). There is no need to modify the other modules of motion-based projection, appearance-based matching, and focus localization in *Robye*.

Limitations of *Robye*: We discuss the potential limitations of the components in *Robye* and provide suggestions on their improvement as follows:

(1) Motion-based projection: The performance of position prediction and tracking affects the accuracy of object projection. For lightweight position prediction and tracking algorithms such as the Kalman Filter (Welch et al. 1995), ORB (Mur-Artal and Tardós 2014), and Optical Flow (Kale et al. 2015), their prediction and tracking accuracies tend to decrease in more dynamic environments. For example, in high-density crowds, an object may not appear in consecutive frames but rather intermittently. In high-speed driving, motion blur requires additional correction in pixel-based tracking. To adapt the motion-based projection to different environments, an environment-type detection module can be incorporated to switch the object projection schemes accordingly. For example, in high-density crowds, tracking can be extended from two consecutive frames to multiple frames. In high-speed driving scenarios, motion blur in RGB frames can be corrected using deblurring methods (Cho and Lee 2009; Shan et al. 2008; Nayar and Ben-Ezra 2004).

(2) Appearance-based matching: The robustness of appearance-based matching depends on the relative displacement between the self robot and other objects and the background, especially for pixel-value comparisons. A large relative displacement can cause a drastic change in the view of the same object, such as shifting from a front view to a side view. In such cases, appearance-based matching fails and the unmatched parts are processed by the DNN-based visual encoder, which results in redundant computation. One potential solution to address these issues is the adoption of Siamese Neural Networks to improve the robustness of appearance-based matching by capturing the semantic meaning (Chicco 2021; Melekhov et al. 2016). When integrating such neural networks into the appearance-based matching, it is important to consider the computational overhead to ensure efficiency.

(3) Focus localization: Although human experts only need to specify the object types to focus on, the training of the focus localization module still relies on a supervised approach using pre-trained object detection models. This dependency on human input limits the generality of the focus localization. One potential improvement is to train the focus localization module in an unsupervised manner. For

example, we can randomly initialize the types of objects to focus and adjust them adaptively based on the application's metrics (*e.g.,* driving score and SPL). Specifically, if the metric improves when a particular type of object is included, it indicates that the object requires focus, and vice versa.

(4) Motion-conditioned visual encoder: The training of the motion-conditioned visual encoder involves the typical retraining process used for compressed DNN models (He et al. 2017a). While it is common to retrain or fine-tune DNN models after compression, it adds extra work to prepare the motion-conditioned visual encoder for a robotic application. Given the trend towards developing foundation models that are generally applicable across different applications (Xu et al. 2024), it is also crucial, in the future work, to explore methods for preparing a foundation motion-conditioned visual encoder that can be utilized in various applications, which avoids the need for retraining from scratch as required by other model compression techniques.

8 Conclusion

This paper proposes *Robye*, a bionic visual encoding framework to tackle the challenge of efficient visual encoding in mobile robotic applications. Drawing inspiration from the human visual system, *Robye* incorporates focus processing and motion-aware content correlation mechanisms into the mobile robotic encoding domain. The dual workflows in our framework include high-resolution focus processing, lower-resolution frame analysis, motion-based projection for cross-frame correlation, and a motion-conditioned adaptive DNN execution strategy. These designs collaboratively lead to a reduction in computational costs associated with encoding tasks. Specifically, our evaluation across robotic scenarios of autonomous driving, vision-language navigation, and drone navigation, demonstrates its capabilities. Specifically, *Robye* outperforms baselines in speed (ranging from 1.2 to 3.3× faster), performance (showing a 4% to 29% increase), and power consumption (−36% to −47%).

Author contributions Xueyu Hou wrote the main manuscript text and prepared Figs. 1, 2,3, 4, 5, 6, 7, and 8. Xueyu Hou and Yongjie Guan prepared figures 9, 10, 11, 12, 13, and 14 and Table 1. All authors reviewed the manuscript.

Data availability No datasets were generated or analysed during the current study.

Declarations

Conflict of interest The authors declare no competing interests.

References

Anderson, P., Wu, Q., Teney, D., Bruce, J., Johnson, M., Sünderhauf, N., Reid, I., Gould, S., Van Den Hengel, A.: Vision-and-language navigation: interpreting visually-grounded navigation instructions in real environments. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pp. 3674–3683 (2018)

Awan, M., Shin, J.: Semantic video segmentation with dynamic key-frame selection and distortion-aware feature rectification. Image Vis. Comput. **110**, 104184 (2021)

Badrinarayanan, V., Kendall, A., Cipolla, R.: Segnet: a deep convolutional encoder-decoder architecture for image segmentation. IEEE Trans. Pattern Anal. Mach. Intell. **39**(12), 2481–2495 (2017)

Bahdanau, D., Cho, K., Bengio, Y.: Neural machine translation by jointly learning to align and translate. arXiv preprint arXiv:1409.0473 (2014)

Becker, M.W., Pashler, H., Anstis, S.M.: The role of iconic memory in change-detection tasks. Perception **29**(3), 273–286 (2000)

Beers, R.J., Sittig, A.C., Gon Denier, J.J.: How humans combine simultaneous proprioceptive and visual position information. Exp. Brain Res. **111**, 253–261 (1996)

Blukis, V., Brukhim, N., Bennett, A., Knepper, R.A., Artzi, Y.: Following high-level navigation instructions on a simulated quadcopter with imitation learning. arXiv preprint arXiv:1806.00047 (2018)

Bolya, D., Zhou, C., Xiao, F., Lee, Y.J.: Yolact: real-time instance segmentation. In: Proceedings of the IEEE/CVF International Conference on Computer Vision, pp. 9157–9166 (2019)

Chen, X., Kundu, K., Zhang, Z., Ma, H., Fidler, S., Urtasun, R.: Monocular 3d object detection for autonomous driving. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pp. 2147–2156 (2016)

Chen, L.-C., Papandreou, G., Schroff, F., Adam, H.: Rethinking atrous convolution for semantic image segmentation. arxiv 2017. arXiv preprint arXiv:1706.05587 (2019)

Chen, Z., Hu, P., Zhang, L., Lu, H., He, Y., Wang, S., Zhang, X., Hu, M., Li, T.: Video object segmentation via structural feature reconfiguration. In: ACCV (2022)

Chicco, D.: Siamese neural networks: an overview. Artif. Neural Netw. **2190**, 73–94 (2021)

Cho, S., Lee, S.: Fast motion deblurring. In: ACM SIGGRAPH Asia 2009 Papers, pp. 1–8 (2009)

Coltheart, M.: Iconic memory and visible persistence. Percept. Psychophys. **27**, 183–228 (1980)

Das, A., Gkioxari, G., Lee, S., Parikh, D., Batra, D.: Neural modular control for embodied question answering. In: Conference on Robot Learning, pp. 53–62. PMLR (2018)

Du, K., Pervaiz, A., Yuan, X., Chowdhery, A., Zhang, Q., Hoffmann, H., Jiang, J.: Server-driven video streaming for deep learning inference. In: Proceedings of the Annual Conference of the ACM Special Interest Group on Data Communication on the Applications, Technologies, Architectures, and Protocols for Computer Communication, pp. 557–570 (2020)

Duan, J., Yu, S., Tan, H.L., Zhu, H., Tan, C.: A survey of embodied AI: from simulators to research tasks. IEEE Trans. Emerg. Top. Comput. Intell. **6**(2), 230–244 (2022)

Fang, K., Toshev, A., Fei-Fei, L., Savarese, S.: Scene memory transformer for embodied agents in long-horizon tasks. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, pp. 538–547 (2019)

Fang, B., Zeng, X., Zhang, M.: Nestdnn: resource-aware multi-tenant on-device deep learning for continuous mobile vision. In: Proceedings of the 24th Annual International Conference on Mobile Computing and Networking, pp. 115–127 (2018)

Fung, A., Benhabib, B., Nejat, G.: Robots autonomously detecting people: a multimodal deep contrastive learning method robust to

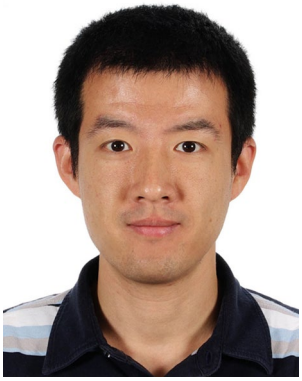
intraclass variations. IEEE Robot. Automat. Lett. **8**(6), 3550–3557 (2023)
Gegenfurtner, K.R., Sperling, G.: Information transfer in iconic memory experiments. J. Exp. Psychol. Hum. Percept. Perform. **19**(4), 845 (1993)
Girshick, R.: Fast r-cnn. In: Proceedings of the IEEE International Conference on Computer Vision, pp. 1440–1448 (2015)
Google: Google Pixel 7. https://store.google.com/product/pixel_7?hl=en-US. Accessed 24 Oct 2023
Gr, O.-J., Gr, U., et al.: Interaction of vestibular and visual inputs in the visual system. Prog. Brain Res. **37**, 573–583 (1972)
Gu, J., Stefani, E., Wu, Q., Thomason, J., Wang, X.E.: Vision-and-language navigation: A survey of tasks, methods, and future directions. arXiv preprint arXiv:2203.12667 (2022)
He, K., Zhang, X., Ren, S., Sun, J.: Deep residual learning for image recognition. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pp. 770–778 (2016)
He, Y., Zhang, X., Sun, J.: Channel pruning for accelerating very deep neural networks. In: Proceedings of the IEEE International Conference on Computer Vision, pp. 1389–1397 (2017a)
He, K., Gkioxari, G., Dollár, P., Girshick, R.: Mask r-cnn. In: ICCV (2017b)
Hu, R., Fried, D., Rohrbach, A., Klein, D., Darrell, T., Saenko, K.: Are you looking? Grounding to multiple modalities in vision-and-language navigation. arXiv preprint arXiv:1906.00347 (2019)
Huang, S., Wang, Z., Li, P., Jia, B., Liu, T., Zhu, Y., Liang, W., Zhu, S.-C.: Diffusion-based generation, optimization, and planning in 3d scenes. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, pp. 16750–16761 (2023)
Intel: RealSense LiDAR Camera L515. https://www.intelrealsense.com/lidar-camera-l515/. Accessed 24 Oct 2023
Jain, S., Wang, X., Gonzalez, J.E.: Accel: a corrective fusion network for efficient semantic segmentation on video. In: ICCV (2019)
Kale, K., Pawar, S., Dhulekar, P.: Moving object tracking using optical flow and motion vector estimation. In: 2015 4th International Conference on Reliability, Infocom Technologies and Optimization (ICRITO)(trends and Future Directions), pp. 1–6. IEEE (2015)
Khandelwal, A., Weihs, L., Mottaghi, R., Kembhavi, A.: Simple but effective: Clip embeddings for embodied AI. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, pp. 14829–14838 (2022)
Kim, D., Woo, S., Lee, J.-Y., Kweon, I.S.: Dense pixel-level interpretation of dynamic scenes with video panoptic segmentation. IEEE Trans. Image Process. **31**, 5383–5395 (2022)
Lee, S.-P., Chen, S.-C., Peng, W.-H.: Gsvnet: guided spatially-varying convolution for fast semantic segmentation on video. In: ICME (2021)
Li, J., Wang, W., Chen, J., Niu, L., Si, J., Qian, C., Zhang, L.: Video semantic segmentation via sparse temporal transformer. In: MM (2021)
Li, J., Tan, H., Bansal, M.: Envedit: Environment editing for vision-and-language navigation. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, pp. 15407–15417 (2022)
Liu, L., Li, H., Gruteser, M.: Edge assisted real-time object detection for mobile augmented reality. In: MobiCom (2019)
Liu, S., Wang, T., Li, J., Sun, D., Srivastava, M., Abdelzaher, T.: Adamask: Enabling machine-centric video streaming with adaptive frame masking for dnn inference offloading. In: Proceedings of the 30th ACM International Conference on Multimedia, pp. 3035–3044 (2022)
Long, J., Shelhamer, E., Darrell, T.: Fully convolutional networks for semantic segmentation. In: CVPR (2015)
Majumdar, A., Yadav, K., Arnaud, S., Ma, J., Chen, C., Silwal, S., Jain, A., Berges, V.-P., Wu, T., Vakil, J., et al.: Where are we in the search for an artificial visual cortex for embodied intelligence? Adv. Neural Inf. Process. Syst. **36**, 655–677 (2024)
Melekhov, I., Kannala, J., Rahtu, E.: Siamese network features for image matching. In: 2016 23rd International Conference on Pattern Recognition (ICPR), pp. 378–383. IEEE (2016)
Misra, D., Bennett, A., Blukis, V., Niklasson, E., Shatkhin, M., Artzi, Y.: Mapping instructions to actions in 3d environments with visual goal prediction. arXiv preprint arXiv:1809.00786 (2018)
Mur-Artal, R., Tardós, J.D.: Orb-slam: tracking and mapping recognizable. In: Proceedings of the Workshop on Multi View Geometry in Robotics (MVIGRO)-RSS (2014)
Murti, C., Narshana, T., Bhattacharyya, C.: Tvsprune-pruning non-discriminative filters via total variation separability of intermediate representations without fine tuning. In: The Eleventh International Conference on Learning Representations (2022)
Nair, S., Rajeswaran, A., Kumar, V., Finn, C., Gupta, A.: R3m: a universal visual representation for robot manipulation. arXiv preprint arXiv:2203.12601 (2022)
Nayar, S.K., Ben-Ezra, M.: Motion-based motion deblurring. IEEE Trans. Pattern Anal. Mach. Intell. **26**(6), 689–698 (2004)
NVIDIA TensorRT. NVIDIA Developer Documentation (2024)
NVIDIA: Jetson AGX Orin. https://www.nvidia.com/en-il/autonomous-machines/embedded-systems/jetson-orin/. Accessed 24 Oct 2023
NVIDIA: Jetson AGX Xavier. https://www.nvidia.com/en-us/autonomous-machines/embedded-systems/jetson-agx-xavier/. Accessed 24 Oct 2023
Pari, J., Shafiullah, N.M., Arunachalam, S.P., Pinto, L.: The surprising effectiveness of representation learning for visual imitation. arXiv preprint arXiv:2112.01511 (2021)
Petrovai, A., Nedevschi, S.: Time–space transformers for video panoptic segmentation. In: ICCV (2022)
Pixhawk Flight Controllor. https://pixhawk.org/. Accessed 24 Oct 2023
Qi, C.R., Yi, L., Su, H., Guibas, L.J.: Pointnet++: deep hierarchical feature learning on point sets in a metric space. Adv. Neural Inf. Process. Syst. **30**, 5099–5108 (2017)
Qi, C.R., Liu, W., Wu, C., Su, H., Guibas, L.J.: Frustum pointnets for 3d object detection from rgb-d data. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pp. 918–927 (2018)
Qi, Y., Pan, Z., Zhang, S., Hengel, A., Wu, Q.: Object-and-action aware model for visual language navigation. In: European Conference on Computer Vision, pp. 303–317. Springer (2020)
Radosavovic, I., Xiao, T., James, S., Abbeel, P., Malik, J., Darrell, T.: Real-world robot learning with masked visual pre-training. In: Conference on Robot Learning, pp. 416–426. PMLR (2023)
Rhee, H., Min, D., Hwang, S., Andreis, B., Hwang, S.J.: Distortion-aware network pruning and feature reuse for real-time video segmentation. arXiv (2022)
Ryu, H., Kim, J., An, H., Chang, J., Seo, J., Kim, T., Kim, Y., Hwang, C., Choi, J., Horowitz, R.: Diffusion-edfs: bi-equivariant denoising generative modeling on se (3) for visual robotic manipulation. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, pp. 18007–18018 (2024)
Samsung: Samsung Galaxy S22. https://www.samsung.com/us/smartphones/galaxy-s22/. Accessed 24 Oct 2023
Schumann, R., Riezler, S.: Analyzing generalization of vision and language navigation to unseen outdoor areas. arXiv preprint arXiv:2203.13838 (2022)
Seong, H., Oh, S.W., Lee, J.-Y., Lee, S., Lee, S., Kim, E.: Hierarchical memory matching network for video object segmentation. In: ICCV (2021)
Shan, Q., Jia, J., Agarwala, A.: High-quality motion deblurring from a single image. ACM Trans. Graph. (TOG) **27**(3), 1–10 (2008)

Shao, H., Wang, L., Chen, R., Li, H., Liu, Y.: Safety-enhanced autonomous driving using interpretable sensor fusion transformer. In: Conference on Robot Learning, pp. 726–737. PMLR (2023)
Simonyan, K., Zisserman, A.: Very deep convolutional networks for large-scale image recognition. arXiv preprint arXiv:1409.1556 (2014)
Stewart, E.E., Valsecchi, M., Schütz, A.C.: A review of interactions between peripheral and foveal vision. J. Vis. **20**(12), 2–2 (2020)
Sun, G., Liu, Y., Ding, H., Probst, T., Van Gool, L.: Coarse-to-fine feature mining for video semantic segmentation. In: ICCV (2022a)
Sun, G., Liu, Y., Tang, H., Chhatkuli, A., Zhang, L., Van Gool, L.: Mining relations among cross-frame affinities for video semantic segmentation. In: ECCV (2022b)
Tan, M., Pang, R., Le, Q.V.: Efficientdet: Scalable and efficient object detection. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, pp. 10781–10790 (2020)
Team, C.: CARLA Autonomous Driving Leaderboard. https://leaderboard.carla.org/. Accessed 5 Nov 2023
Thomason, J., Gordon, D., Bisk, Y.: Shifting the baseline: Single modality performance on visual navigation & qa. arXiv preprint arXiv:1811.00613 (2018)
Wahid, A., Stone, A., Chen, K., Ichter, B., Toshev, A.: Learning object-conditioned exploration using distributed soft actor critic. In: Conference on Robot Learning, pp. 1684–1695. PMLR (2021)
Wang, X., Zhang, R., Kong, T., Li, L., Shen, C.: Solov2: dynamic and fast instance segmentation. Adv. Neural. Inf. Process. Syst. **33**, 17721–17732 (2020)
Wang, H., Jiang, X., Ren, H., Hu, Y., Bai, S.: Swiftnet: Real-time video object segmentation. In: ICCV (2021)
Wang, H., Tan, A.H., Nejat, G.: Navformer: a transformer architecture for robot target-driven navigation in unknown and dynamic environments. IEEE Robot. Automat. Lett. (2024). https://doi.org/10.1109/LRA.2024.3412638
Welch, G., Bishop, G., et al.: An introduction to the Kalman filter (1995)
Wijmans, E., Datta, S., Maksymets, O., Das, A., Gkioxari, G., Lee, S., Essa, I., Parikh, D., Batra, D.: Embodied question answering in photorealistic environments with point cloud perception. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, pp. 6659–6668 (2019a)
Wijmans, E., Kadian, A., Morcos, A., Lee, S., Essa, I., Parikh, D., Savva, M., Batra, D.: Dd-ppo: Learning near-perfect pointgoal navigators from 2.5 billion frames. arXiv preprint arXiv:1911.00357 (2019b)
Woo, S., Kim, D., Lee, J.-Y., Kweon, I.S.: Learning to associate every segment for video panoptic segmentation. In: ICCV (2021)
Wortsman, M., Ehsani, K., Rastegari, M., Farhadi, A., Mottaghi, R.: Learning to learn how to learn: Self-adaptive visual navigation using meta-learning. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, pp. 6750–6759 (2019)
Xie, Q., Lai, Y.-K., Wu, J., Wang, Z., Zhang, Y., Xu, K., Wang, J.: Mlcvnet: Multi-level context votenet for 3d object detection. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, pp. 10447–10456 (2020)
Xie, H., Yao, H., Zhou, S., Zhang, S., Sun, W.: Efficient regional memory network for video object segmentation. In: ICCV (2021)
Xu, Z., Wu, K., Wen, J., Li, J., Liu, N., Che, Z., Tang, J.: A survey on robotics with foundation models: toward embodied AI. arXiv preprint arXiv:2402.02385 (2024)
Yang, B., Wang, J., Clark, R., Hu, Q., Wang, S., Markham, A., Trigoni, N.: Learning object bounding boxes for 3d instance segmentation on point clouds. Adv. Neural Inf. Process. Syst. **32** (2019)
Yang, K., Yi, J., Lee, K., Lee, Y.: Flexpatch: fast and accurate object detection for on-device high-resolution live video analytics. In: IEEE INFOCOM 2022-IEEE Conference on Computer Communications, pp. 1898–1907. IEEE (2022)
Ye, J., Batra, D., Wijmans, E., Das, A.: Auxiliary tasks speed up learning point goal navigation. In: Conference on Robot Learning, pp. 498–516. PMLR (2021)
Ye, W., Lan, X., Su, G., Bao, H., Cui, Z., Zhang, G.: Hybrid tracker with pixel and instance for video panoptic segmentation. arXiv (2022)
Yen-Chen, L., Zeng, A., Song, S., Isola, P., Lin, T.-Y.: Learning to see before learning to act: Visual pre-training for manipulation. In: 2020 IEEE International Conference on Robotics and Automation (ICRA), pp. 7286–7293. IEEE (2020)
Zhang, Y., Tan, H., Bansal, M.: Diagnosing the environment bias in vision-and-language navigation. arXiv preprint arXiv:2005.03086 (2020)
Zhang, Z., Liniger, A., Dai, D., Yu, F., Van Gool, L.: End-to-end urban driving by imitating a reinforcement learning coach. In: Proceedings of the IEEE/CVF International Conference on Computer Vision, pp. 15222–15232 (2021)
Zhou, Y., Zhang, H., Lee, H., Sun, S., Li, P., Zhu, Y., Yoo, B., Qi, X., Han, J.-J.: Slot-vps: Object-centric representation learning for video panoptic segmentation. In: ICCV (2022)
Zhu, Y., Zhu, F., Zhan, Z., Lin, B., Jiao, J., Chang, X., Liang, X.: Vision-dialog navigation by exploring cross-modal memory. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, pp. 10730–10739 (2020)
Zhu, W., Qi, Y., Narayana, P., Sone, K., Basu, S., Wang, X.E., Wu, Q., Eckstein, M., Wang, W.Y.: Diagnosing vision-and-language navigation: What really matters. arXiv preprint arXiv:2103.16561 (2021)
Zhuang, J., Wang, Z., Wang, B.: Video semantic segmentation with distortion-aware feature correction. IEEE Trans. Circuits Syst. Video Technol. **31**(8), 3128–3139 (2020)
Zhuang, J., Wang, Z., Li, J.: Video semantic segmentation with inter-frame feature fusion and inner-frame feature refinement. arXiv (2023)

Xueyu Hou is an Assistant Professor in the Department of Electrical and Computer Engineering at University of Maine. She received her Ph.D. degree in the Electrical and Computer Engineering Department at the New Jersey Institute of Technology (NJIT). Before coming to NJIT, she obtained her B.S. and M.S. degree in Electrical Engineering from Xi'an Jiaotong University. She was also a student in the Special Class of Gifted Young in Xi'an Jiaotong University. Dr. Hou is the recipient of the NJIT Hashimoto Prize 2024. Her current research interests include efficient artificial intelligence, humancentered computing, mobile edge computing, and sustainable computing.

Yongjie Guan is an Assistant Professor in the Department of Electrical and Computer Engineering at University of Maine. He received his Ph.D. and Master degree in the Electrical and Computer Engineering Department at the New Jersey Institute of Technology (NJIT). Before coming to NJIT, he obtained his B.S. degree in Electrical Engineering from University of Electronic Science and Technology of China. His current research interests include mobile X reality System, mobile edge computing, unmanned aircraft systems, and humancentered computing.

Tao Han is an Associate Professor in the Department of Electrical and Computer Engineering at New Jersey Institute of Technology (NJIT) and an IEEE Senior Member. Before joining NJIT, Dr. Han was an Assistant Professor in the Department of Electrical and Computer Engineering at the University of North Carolina at Charlotte. Dr. Han received his Ph.D. in Electrical Engineering from NJIT in 2015 and is the recipient of the NSF CAREER Award 2021, the Newark College of Engineering Outstanding Dissertation Award 2016, the NJIT Hashimoto Prize 2015, and the New Jersey Inventors Hall of Fame Graduate Student Award 2014. His papers win the IEEE International Conference on Communications (ICC) Best Paper Award 2019 and IEEE Communications Society's Transmission, Access, and Optical Systems (TAOS) Best Paper Award 2019. His research interests include mobile edge computing, machine learning, mobile X reality, 5G system, Internet of Things, and smart grid.



Cong Wang is currently an Associate Professor in the ECE department at New Jersey Institute of Technology. Before joining NJIT in 2015, Dr. Wang was a Lecturer and Research Engineer at University of California, Berkeley. He obtained his PhD degree in the area of Controls and Dynamics from UC Berkeley in 2014, before which he attended Tsinghua University and obtained his master's degree in Automotive Engineering and bachelor's degree in Manufacturing Engineering and Automation in 2010 and 2008 respectively.