Infrastructure-Guided Connectivity-Enhanced Road Crack Detection and Estimation

Haosong Xiao*, Yamini Ramesh**, Rishabh Shukla**, Swarat Sarkar ** and Chaozhe R. He*
*Department of Mechanical and Aerospace Engineering
**School of Engineering and Applied Science
University at Buffalo (SUNY), Buffalo, New York, USA
{haosongx, yaminira, rshukla3, swaratsa, chaozheh}@buffalo.edu

***Abstract*—In this paper, we report the world's first infrastructure-guided communication-enhanced road crack detection pipeline that is effective and implementable on passenger vehicles. We first design a customized communication protocol to transmit the region of interest from the infrastructure to the vehicle. With proper camera image processing (e.g., dynamic cropping and frame selection), the focused images are provided to the crack detection model. Leveraging state-of-the-art crack detection model backbones and a carefully prepared dataset comprising a forward-facing view with a crack, we train the model to improve crack-detection performance. We demonstrate the full detection pipeline on an experimental vehicle platform, showcase the detection effectiveness, and project future research directions.**

***Index Terms*—Connected vehicles, Smart infrastructure, Vehicle Environment Perception**

I. Introduction

Infrastructure maintenance will cost trillions of US dollars in investment between 2024 and 2033 [1]. Among various categories of infrastructure, bridges, and more specifically, bridge decks face a significant funding gap for maintenance. Bridge decks account for a significant portion of overall bridge maintenance costs and directly affect safety and user experience [2]–[4]. It is estimated that 50% to 85% of total bridge maintenance costs are attributed to deck maintenance [5]. Deterioration mechanisms such as reflective cracking, joint spalling, and chloride-induced corrosion are well documented, particularly in precast decks with multiple panel joints [6], [7]. Early detection of these defects is crucial: once rebar corrosion and delamination advance, repairs become far more expensive and disruptive. Visible signs of damage from the surface, such as joint cracking, grout spalling, or reflective cracking, can indicate underlying issues, and could lead to structural degradation if unchecked. And overly aggressive cracking would require repairs which could lead to road closure, disrupting traffic [8].

Federal regulations mandate routine inspections every 24 months to ensure safety. Manual inspections remain the predominant means of inspection for professional tasks requiring detailed training [9], leading to steep labor costs, traffic disruptions, and safety risks [10]. Therefore, developing effective and efficient crack-detection approaches for the bridge surface inspection, or more general road surface crack detection, is crucial for maintaining bridge health in an efficient and economical manner. While remote sensing systems using drones and dedicated inspection vehicles are commercially available, their operating costs remain high, making continuous monitoring uneconomical [11], [12]. Currently, there is a lack of cost-effective, automated inspection systems capable of detecting damage between routine inspections, particularly early signs of degradation. Developing such a system would enable continuous monitoring, reducing the risk of undetected damage and enhancing the overall durability of precast bridge components.

With the emerging vehicular technologies in sensing and connectivity, modern vehicles are increasingly capable of perceiving road conditions. Harnesses crowd sensing capability brought by road vehicles equipped with front-view cameras could be used to build a cost-effective, continuous bridge surface crack inspection system. Nationwide deployment plan of Vehicle-to-everything (V2X) devices [13] facilitates the feasibility of effective crowd sensing, further enabling such system. Built on these new technologies, in this paper, we report our efforts towards such inspection system by making the following contributions:

1) We establish the world's first infrastructure-guided connectivity-enhanced crack detection framework that leverages all road vehicles for road surface inspections
2) We provide a detailed design recipe for communication protocol design, crack detection model tuning, camera calibration, and crack calculation.
3) We release the first dataset of surface cracks from the vehicle's front-view camera.
4) We implement and demonstrate the designed crack detection framework on a real-vehicle platform.

II. Background

In this section, we first survey related work in the field. With a motivating example, we then highlight the limitations of existing work and preview the design in this work.

The source code and data is available at https://github.com/CHELabUB/cav_road_crack_detection

A. Related Work

Prior crack detection research can be divided into two categories. One focuses on improving the crack recognition accuracy by advancing the detection and segmentation models. Early approaches primarily relied on low-level image intensity analysis, using statistical analysis of pixels' surrounding intensity difference to recognize cracks from the pavement surface [14]. Kernel-based edge detectors, such as Sobel and Canny operators, were also widely used [15], [16]. These methods are sensitive to surface texture and noise due to lighting conditions. More recent techniques, including shadow moving and tensor voting, are used to reduce the impact of image noise [17]. The extensive use of parameters increase the effort of hand-tuning and limits generalization capabilities. These limitations can be mitigated by adopting learning-based approaches, which train neural networks on annotated pavement images to learn different crack features, thereby improving the robustness of the detection performance under illumination variation and surface texture variations [18]–[20].

The other focuses on implementing detection models on existing and purposefully developed unmanned aerial and ground vehicles (UAVs and UGVs) to streamline the detection process. Prior works using UAVs [21], [22] are constrained by the limited battery life, and the hidden trade-off between flight height and image resolution. A purposefully designed platform combining UGV and robotic arm used in [23] demonstrates strong performance detection performance on daily used traffic corners, but it also raises concern on disrupting traffic during regular detection task. Using a designated road vehicle with detection externally mounted cameras, as shown in [24], can mitigate the disturbance to traffic. Yet its detection range may be limited due to a fixed down-view camera angle and potentially strong requirements on the vehicle's maneuver (e.g., path and speed) for designated crack locations. While front-view camera-based [25] and modification-free vehicle-based [26] platforms demonstrate the potential of autonomous vehicles for crack detection, the crack detection only focused on object detection rather than length estimation making it insufficient for bridge inspection usage.

Modern vehicles with sensors such as front-view cameras could continuously monitor the road, allowing vehicles to assess the status of the infrastructure using vision. Existing work along this direction though focuses only on benefiting the vehicle, e.g., estimating road roughness and friction coefficient and detecting pavement cracks and potholes to improve ride comfort [27], [28]. Opportunities to extend such capability to improve infrastructure durability (e.g., bridge) are unexplored [29]. One obstacle is enabling direct information sharing between vehicles and infrastructure regarding inspection results acquired by road vehicles. With the increase level of connectivity [13] in transportation system, such obstacle can be overcome. Yet challenges remains on how to effectively adapt vehicle centric sensors and algorithm design towards infrastructure inspection applications.

Overall, there remains a gap in developing inspection approaches that enable continuous crack detection on pavement without disrupting traffic or limiting operational time, while maintaining robust detection accuracy.

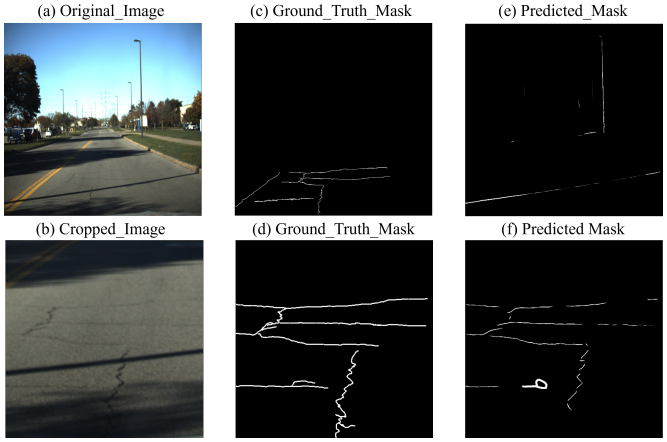


Fig. 1. Dynamic cropping impact: (a) Original full-frame image (2064×1544), (b) Cropped 512×512 region, (c) Ground truth mask of the original frame, (d) Corresponding ground truth of the cropped frame, (e) Predicted mask showing false positives from background noise, (f) Cleaner prediction with precision improvement from 0.20 to 0.70.

TABLE I
DETECTION METRICS: FULL FRAME VS. CROPPED

Metric	Full Frame	Cropped	Improvement
ODS F1	0.2251	0.5288	+135%
Precision	0.2009	0.6966	+247%
Recall	0.3554	0.4780	+34%
OIS F1	0.2688	0.5336	+99%
AP	0.0382	0.3521	+822%

B. Motivating Connectivity and Dynamic Cropping

Despite extensive research on automated crack detection [30], benchmarks are primarily reported on limited datasets, and real-world operational performance remains limited. Existing methods often excel on curated datasets but struggle with the variable conditions encountered in real-world deployment. Real-world crack detection faces a fundamental challenge: full-frame processing of high-resolution front-view camera images contains substantial irrelevant background information that degrades model performance. To showcase this limitation and motivate our designs, we ran the developed crack detection model (details to be provided in Section IV) to both full frame and properly cropped image. We evaluate detection performance using standard segmentation metrics, including Optimal Dataset Scale F1-score (ODS F1), Optimal Image Scale F1-score (OIS F1), precision, recall, and Average Precision (AP), comparing full 2064×1544 frames against cropped regions. ODS F1 and OIS F1 denote the best F1-score over a fixed dataset-wide threshold and per-image adaptive thresholds, respectively, while AP summarizes precision–recall performance across all thresholds. As shown in Fig. 1, full-frame inference yields low precision (0.20) and ODS F1

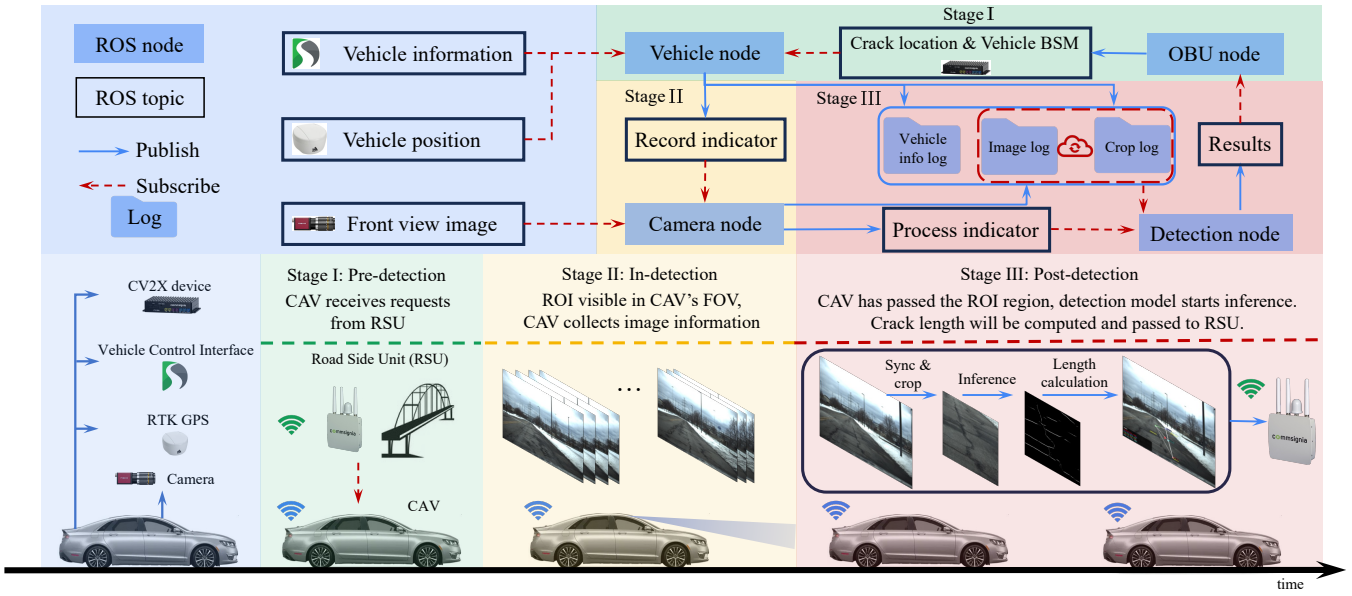


Fig. 2. Design overview of the connectivity-enhanced crack detection pipeline.

(0.23), as background textures, shadows, and road markings introduce false positives.

Motivated by this finding, we design a dynamic cropping algorithm in Section III-A that captures such region of interest (ROI) and significantly reduces background interference. The algorithm is guided by infrastructure through connectivity, to compute an optimal cropping window centered on the target crack using the coordinate transformation pipeline described. Results in Table I show improvements of 247% in precision (from 0.20 to 0.70) and 135% in ODS F1 (from 0.23 to 0.53), highlighting the effectiveness of the proposed approach.

C. Design Overview

Our connectivity-guided crack detection pipeline, as shown in Fig. 2, provides a novel, comprehensive, and extensible framework for autonomous road surface inspection. The pipeline consists of three stages. 1) Pre-detection stage where communication happens between the infrastructure (e.g. bridge) and the vehicles via the cellular-vehicle-to-everything (C-V2X) protocol [31] (c.f., Stage I in Fig. 2). 2) In-detection stage: an adjustable detection range is determined ensures optimal crack resolution using real-time onboard vehicle sensor data (c.f., Stage II in Fig. 2). 3) Post-Detection stage: an improved crack detection model operates on a crack-centered cropped image from the synchronization mechanism ((c.f., Stage III in Fig. 2) and delivers detection results.

The remainder of this paper elaborates on the details of various components: the mathematical formulation enabling accurate crop localization and subsequent crack length estimation in Section III; model training in Section IV; full pipeline implementation and performance on a connected automated vehicle (CAV) platform in Section V.

III. CV2X-BASED VIEW SELECTION

In this section, we first describe the algorithm that determine the focus area around the target crack on the road surface. Then we derive the crack length estimation algorithm based on vehicle location and target crack location. These two algorithms are crucial parts of Stage II and III in Fig. 2

A. Dynamic Cropping Window Design

We assume that infrastructure has a known region of interest (ROI), whose center may contains cracks that critical to structural health. The vehicle receives the GPS coordinates of the region of interest from infrastructure through connectivity. Given the GPS coordinates of ROI, along with the vehicle's coordinate acquired from GPS (e.g., RTK GPS), we can convert the crack's relative position with respect to the vehicle into local east-north-up (ENU) coordinates, and eventually acquire the relative position of the crack with respect to the camera lens denoted as $\mathbf{p}'_{\text{crack2C}}$.

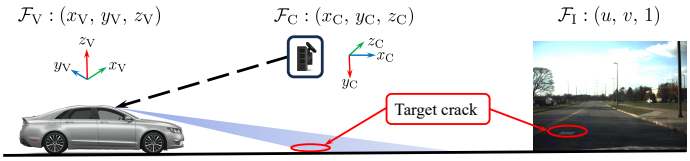


Fig. 3. Vehicle frame to image frame transformation

To determine cropping window location from the full image, the center of the crack is first projected from the 3D vehicle coordinate frame, $\mathcal{F}_V : (x_V, y_V, z_V)$, to the camera coordinate frame $\mathcal{F}_C : (x_C, y_C, z_C)$, and finally to the 2D image frame, $\mathcal{F}_I : (u, v)$, using pinhole camera model. With the converted crack center pixel, the cropping window can be determined while remaining in the camera's field of view

(FOV). In Fig. 3, all these frames are illustrated. Assumed that the crack lies on a flat road surface, the transformation from $\mathcal{F}_{\mathrm{V}} : (x_{\mathrm{V}}, y_{\mathrm{V}}, z_{\mathrm{V}})$ to $\mathcal{F}_{\mathrm{C}} : (x_{\mathrm{C}}, y_{\mathrm{C}}, z_{\mathrm{C}})$ is expressed as

$$\mathbf{p}_{\mathrm{C}} = R_{zyx} R_{\mathrm{C}}^{V} \, \mathbf{p}'_{\mathrm{crack2C}}, \tag{1}$$

where R_{C}^{V} denotes the fixed rotation from $\mathcal{F}_{\mathrm{V}}$ to $\mathcal{F}_{\mathrm{C}}$, and $R_{zyx} = R_z(\phi_{\mathrm{C}})R_x(\theta_{\mathrm{C}})R_y(\psi_{\mathrm{C}})$ represents the camera orientation defined by yaw ϕ_{C}, pitch θ_{C}, and roll ψ_{C}.

Finally, to project the 3D crack center coordinate $\mathbf{p}_{\mathrm{C}}$ onto the image plane $\mathcal{F}_{\mathrm{I}}$, the intrinsic matrix K of the camera is used :

$$\begin{bmatrix} u_{\mathrm{C}} \\ v_{\mathrm{C}} \\ 1 \end{bmatrix} = \frac{1}{z_{\mathrm{C}}} K \, \mathbf{p}_{\mathrm{C}}. \tag{2}$$

An ROI of size $(B_{\mathrm{w}}, B_{\mathrm{h}})$ can be acquired with cropping corners

$$\begin{aligned} u_{\min} &= \max\left(0,\ \min\left(u_{\mathrm{C}} - \frac{B_{\mathrm{w}}}{2},\ I_{\mathrm{w}} - B_{\mathrm{w}}\right)\right), \\ v_{\min} &= \max\left(0,\ \min\left(v_{\mathrm{C}} - \frac{B_{\mathrm{h}}}{2},\ I_{\mathrm{h}} - B_{\mathrm{h}}\right)\right), \\ u_{\max} &= u_{\min} + B_{\mathrm{w}}, \quad v_{\max} = v_{\min} + B_{\mathrm{h}}. \end{aligned} \tag{3}$$

We remark that the proposed design guarantees that for a perfectly calibrated camera, the true crack will always be inside the cropping window. While (3) gives a unique cropping window for a given frame, it is possible that for the same crack, multiple cropped images may be acquired. In Section V-B we describe how the optimal cropping window with the best possible resolution may be selected; c.f., Stage III in Fig. 2.

B. Crack Length Estimation

In this work, we seek to estimate crack length by computing the Euclidean distance between the two reconstructed endpoint points of the crack, expressed in the vehicle coordinate frame. The reconstruction is performed under the assumption that the crack lies on the flat road. We adopt this definition, consistent with the manual inspection instructions [9], and to mitigate the zig-zag nature of fine cracks.

The accuracy of the length estimation depends highly on the camera extrinsic parameters, particularly the camera's mounting yaw ϕ_{C}, pitch θ_{C}, and roll ψ_{C}, as these parameters directly determines the conversion between the vehicle and the camera coordinate frame. Thus, we first design a streamlined calibration of camera extrinsic parameters, specific to crack detection. Then we can proceed to proper length calculation.

1) Camera Calibration: Traditional camera calibration is done by formulating the calculation of intrinsic and extrinsic parameters as an optimization problem, where the objective function minimizes the reconstruction error between observed image points and their predicted projections computed from multiple viewpoints using the tunable camera parameters [32]. While the intrinsic parameters of the camera can be retrieved the manufacturer, the extrinsic parameters remain unknown. Calibrating vehicle camera extrinsic parameters could be challenging due to vehicle motions. In this work, we use a known

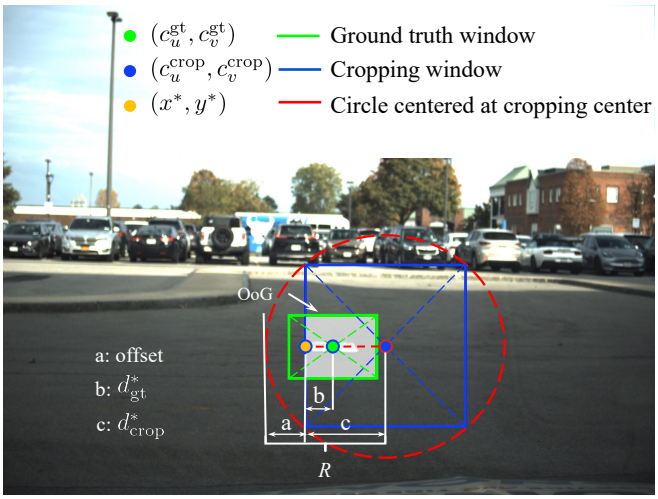


Fig. 4. Camera calibration sample with key quantities used in the algorithm

calibration target with its center GPS position together with the known camera intrinsics. A checkerboard is chosen as the target due to its simplicity and reliability in pattern recognition practice. We further design the size of the cropping windows $(B_{\mathrm{w}}, B_{\mathrm{h}})$ to be big enough to cover the target object in a square box ($B_{\mathrm{w}} = B_{\mathrm{h}}$). Leveraging object detection model YOLO [33], accurate checkerboard bounding boxes can be obtained. We denote the center pixel of the ground truth window and the cropping window as $(c_u^{\mathrm{gt}}, c_v^{\mathrm{gt}})$ and $(c_u^{\mathrm{crop}}, c_v^{\mathrm{crop}})$ respectively. To align the cropped window with the ground truth window as close as possible, we use the unknown extrinsic as the optimization variable and an objective function with two metrics that can effectively quantify the alignment between the cropping windows : overlapping with the ground truth (OoG) and the alignment of traveling distance (AoT). The corresponding definitions are illustrated in Fig. 4 and is detailed as follows.

Overlapping of Ground Truth (OoG): OoG measures ratio of ground truth B^{gt} has been included in the cropping window B.

$$\mathrm{OoG}(c_u^{\mathrm{crop}}, c_v^{\mathrm{crop}}) = \frac{|B \cap B^{\mathrm{gt}}|}{B^{\mathrm{gt}}}, \quad \mathrm{OoG} \in [0, 1]. \tag{4}$$

Alignment of Traveling Distance (AoT): AoT measures the relative distance between two cropping windows. For a square cropping window, the diagonal length can be interpreted as the diameter of a circle centered around the window center $(c_u^{\mathrm{crop}}, c_v^{\mathrm{crop}})$, with radius R. The cropping window center corresponds to the theoretical projected pixel of the crack center. Since the ground true crack is always within the cropping box during calibration by design, one has $(c_x^{\mathrm{gt}}, c_y^{\mathrm{gt}}) \in \left[c_x^{\mathrm{crop}} - \frac{L_{\mathrm{crop}}}{2}, c_x^{\mathrm{crop}} + \frac{L_{\mathrm{crop}}}{2}\right]$. The circle region can then be naturally used as a reference for evaluating the spatial relation between $(c_u^{\mathrm{crop}}, c_v^{\mathrm{crop}})$ and $(c_u^{\mathrm{gt}}, c_v^{\mathrm{gt}})$. A pixel, (x^*, y^*), on the edge of the cropping window that lies on the line connecting the two center pixels and shares the same slope

can be computed as:

$$\begin{aligned} d_{cx} &= \mathbf{max}(c_u^{\mathrm{gt}} - c_u^{\mathrm{crop}}, \epsilon), \\ d_{cy} &= \mathbf{max}(c_v^{\mathrm{gt}} - c_v^{\mathrm{crop}}, \epsilon), \\ s &= \mathbf{min}\left(\frac{\frac{L_{\mathrm{crop}}}{2}}{|d_{cx}|}, \frac{\frac{L_{\mathrm{crop}}}{2}}{|d_{cy}|}\right), \\ (x^*, y^*) &= (c_u^{\mathrm{crop}}, c_v^{\mathrm{crop}}) + s\,(d_{cx}, d_{cy}). \end{aligned} \tag{5}$$

The corresponding distances between the projected edge pixel and the cropping window center, as well as between the projected edge pixel and the ground-truth center, are computed as

$$\begin{aligned} d^*_{\mathrm{crop}} &= \sqrt{\left((x^*)^2 - (c_u^{\mathrm{crop}})^2\right) + \left((y^*)^2 - (c_v^{\mathrm{crop}})^2\right)}, \\ d^*_{\mathrm{gt}} &= \sqrt{\left((x^*)^2 - \left(c_u^{\mathrm{gt}}\right)^2\right) + \left((y^*)^2 - \left(c_v^{\mathrm{gt}}\right)^2\right)}. \end{aligned} \tag{6}$$

The offset between the cropping window boundary and the circle shared the same center is then defined as:

$$\mathrm{offset} = \frac{L_{\mathrm{crop}}}{2} - d^*_{\mathrm{crop}}. \tag{7}$$

The Alignment of Traveling Distance is finally defined as

$$\mathrm{AoT} = \frac{d^*_{\mathrm{gt}} + \mathrm{offset}}{R} \tag{8}$$

A higher AoT indicates that the ground-truth center $(c_u^{\mathrm{gt}}, c_v^{\mathrm{gt}})$ is closer to the cropping window center $(c_u^{\mathrm{crop}}, c_v^{\mathrm{crop}})$.

With these two metrics, the full optimization can be set up as:

$$\begin{aligned} \max_{\phi_{\mathrm{C}}, \theta_{\mathrm{C}}, \psi_{\mathrm{C}}} \quad & \mathrm{OoG}(c_u^{\mathrm{crop}}, c_v^{\mathrm{crop}}) + \mathrm{AoT}(c_u^{\mathrm{crop}}, c_v^{\mathrm{crop}}) \\ \text{s.t.} \quad & \left|c_u^{\mathrm{gt}} - c_u^{\mathrm{crop}}\right| \le \frac{L_{\mathrm{crop}}}{2}, \\ & \left|c_v^{\mathrm{gt}} - c_v^{\mathrm{crop}}\right| \le \frac{L_{\mathrm{crop}}}{2}, \\ & 0 \le \mathrm{OoG}(c_u^{\mathrm{crop}}, c_v^{\mathrm{crop}}) \le 1. \end{aligned} \tag{9}$$

In this work we approximate the solution to (9) using the bisection method with a resolution of 1 degree.

2) Real-time length calculation: To accurately estimate the length between the crack endpoints, an edge-corner detection algorithm is developed. The algorithm takes the crack mask image as input and details on how the crack mask is generated is detailed in Section IV. It extracts pixels that are put crack mask and stays within the crack region. Each extracted pixel is then reconstructed from the image coordinate frame to the vehicle coordinate frame. This reconstruction follows the reverse calculation from (2), with intrinsic matrix K given by

$$\begin{bmatrix} f_x & 0 & c_x \\ 0 & f_y & c_y \\ 0 & 0 & 1 \end{bmatrix}, \tag{10}$$

where f_x and f_y denote the focal length along the camera axes x_{C} and y_{C}, c_x and c_y denote the central pixel in $\mathcal{F}_{\mathrm{I}}$. We first project the 2D image coordinate back to a normalized 3D camera frame $\mathcal{F}_{\mathrm{C}} : (x_{\mathrm{C}},\, y_{\mathrm{C}},\, z_{\mathrm{C}})$,

$$\mathbf{C}_{\mathrm{norm},\mathcal{F}_{\mathrm{C}}} = \begin{bmatrix} \frac{u - c_x}{f_x} & \frac{v - c_y}{f_y} & 1 \end{bmatrix}^{\mathrm{T}}. \tag{11}$$

The normalized coordinates in the camera frame are then transformed into the vehicle frame $\mathcal{F}_{\mathrm{V}} : (x_{\mathrm{V}},\, y_{\mathrm{V}},\, z_{\mathrm{V}})$ using:

$$\mathbf{C}_{\mathrm{norm},\mathcal{F}_{\mathrm{V}}} = (R_{\mathrm{C}}^{\mathrm{V}})^{\mathrm{T}}\, R_{zyx}^{\mathrm{T}}\, \mathbf{C}_{\mathrm{norm},\mathcal{F}_{\mathrm{C}}} \tag{12}$$

We remark that the reconstructed pixel is normalized so it must be scaled back to be accurately expressed in the vehicle frame. Recall the camera offset and the assumption that the cracks lie on the flat ground surface, the de-normalized coordinates are obtained using:

$$\mathbf{C}_{\mathcal{F}_{\mathrm{V}}} = \mathbf{t}_{\mathrm{VC}} + C_z\, \mathbf{C}_{\mathrm{norm},\mathcal{F}_{\mathrm{V}}} \tag{13}$$

$$C_z = -\frac{(\mathbf{t}_{\mathrm{VC}})_z}{(\mathbf{C}_{\mathrm{norm},\mathcal{F}_{\mathrm{V}}})_z}, \tag{14}$$

where $\mathbf{t}_{\mathrm{VC}}$ denotes the (assumed known) offset of the camera lens with respect to the RTK GPS antenna, expressed in the vehicle frame. The reconstructed points are subsequently divided into four quadrants with respect to the crack center, and edge locations are selected as the pixels with the maximum Euclidean distance from the center within each quadrant. The edge selection algorithm is summarized in Algorithm 1.

Algorithm 1 Edge Corner Selection

Require: Highlighted pixel set $\{x_k\}_{k=1}^{N}$
Ensure: Quadrant correlation vector Corr, edge coordinates $\{E_q\}$

```
Extract N highlighted pixels
for k = 1 to N do
    Transform x_k to vehicle-frame coordinates
end for
Compute center coordinate x_C of {x_k}
Initialize dictionary E ← ∅
Initialize vector Corr ∈ R^4
for q = 1 to 4 do
    X_q ← {x_k | x_k ∈ quadrant q}
    if X_q ≠ ∅ then
        x_k^* ← arg max_{x_k ∈ X_q} ||x_k − x_C||_2
        E_q ← {x_k^*}
        Corr_q ← 1
    else
        Corr_q ← 0
    end if
end for
```

IV. Crack Detection Model

In this section, we describe the adaptation of the existing crack detection models for real-time images from vehicle's front view camera. In this work, we focus on improving the generalization under real-world operational condition. We considered two leading open-source semantic segmentation architectures for crack detection given their balanced accuracy

and real-time performance [34]: DeepCrack, a multi-scale convolution encoder-decoder model [18], is employed using its pre-trained weights without modifications; LECSFormer, a transformer-based segmentation based model [19], is adapted and improved for our real-time detection pipeline.

A. Model Reconstruction

To fill missing details and recover baseline model weight for standard dataset, we reconstruct and further refine the LECSFormer training framework, with focuses on loss function and training data augmentation.

1) Loss function: One major difficulty in crack segmentation is the class imbalance, where crack pixels only take a small fraction of the image. To address this, the original LESCFormer uses Binary Cross-Entropy (BCE) loss with high penalty weight to the crack pixels using $\omega_{\text{pos}} = 5.0$ and $\omega_{\text{neg}} = 1.0$:

$$\mathcal{L}_{\text{wBCE}} = \frac{1}{N}\sum_{i=1}^{N} \omega_i \Big[-y_i \log \sigma(\hat{y}_i) - (1-y_i) \log\big(1-\sigma(\hat{y}_i)\big)\Big], \tag{15}$$

where $\hat{y}_i$ denotes the predicted logits, $y_i \in \{0,1\}$ is the ground truth, $\sigma(\cdot)$ is the sigmoid function, and ω_i is selected based on class label. To further optimize region-level overlap, we includes the Dice loss:

$$\mathcal{L}_{\text{Dice}} = 1 - \frac{2\sum_{i=1}^{N} \sigma(\hat{y}_i) y_i + \epsilon}{\sum_{i=1}^{N} \sigma(\hat{y}_i) + \sum_{i=1}^{N} y_i + \epsilon}. \tag{16}$$

We use $\epsilon = 1.0$ for numerical stability. The final loss is a weighted combination of both terms:

$$\mathcal{L}_{\text{combined}} = \lambda_{\text{BCE}} \mathcal{L}_{\text{wBCE}} + \lambda_{\text{Dice}} \mathcal{L}_{\text{Dice}} \tag{17}$$

where $\lambda_{\text{BCE}} = 0.7$ and $\lambda_{\text{Dice}} = 0.3$ are used.

Following the original multi-scale supervision strategy in [19], this combined loss is applied to the final output and all intermediate decoder outputs:

$$\mathcal{L}_{\text{total}} = \mathcal{L}^{\text{out}}_{\text{combined}} + \sum_{k=1}^{K} \mathcal{L}^{\text{mid},k}_{\text{combined}}, \tag{18}$$

where K denotes the number of supervised decoder stages. This formulation improves robustness to class imbalance while preserving multi-scale feature consistency.

2) Image augmentation: Transformer based models such as LECSFormer require large and diverse training data to generalize effectively, but the existing crack datasets are suffered from the size and inconsistent visibility. To address this, we design a crack-density-aware augmentation strategy, in which training samples are categorized into four groups (none, minimal, moderate, dense) based on the number of crack appearances in each sample image. Categories with fewer samples are considered as underrepresented, and will be augmented using geometric transformation, photometric variation, elastic deformation and density-aware cropping to upscale their representation by a factor of 2.

TABLE II
BENCHMARK PERFORMANCE ON CRKWH100 DATASET

Model	Train Data	ODS F1	AP
LECSFormer [19]	CT260+CLS315	0.9530	Not-provided
LECSFormer (Ours)	CT260	0.9184	0.4973

TABLE III
PERFORMANCE ON REAL-TIME FRONT VIEW CAMERA DATASET (RT100)

Model	ODS F1	Precision	Recall	AP
LECSFormer	0.3744	0.3425	0.4605	0.1371
DeepCrack	0.3374	0.2912	0.4359	0.1225

3) Reconstruction results: To evaluate our reconstructed LECSFormer model training pipeline performance, we retrain the model on the augmented CrackTree260 dataset and further benchmark performance on the CRKWH100 dataset (used in the literature to check model generalization capabilities [34]). The reproduced benchmark results, together with the reported metrics from the original work, are summarized in Table II. Overall, we are able to reproduce similar performance, despite using different augmentation methods.

B. Front View Camera Image Performance

To evaluate real-world performance from the vehicle's front view camera images, we collect a dataset of 443 images recorded under diverse operational conditions, including different lighting, glare, motion blur, and asphalt textures. The dataset is split into 343 training images (denoted as RT 343) and 100 testing images (denoted as RT 100). We adopt a hybrid annotation pipeline to accelerate the training set annotation: in addition to manual annotation, we used black hat filter [32] to generate initial crack mask, and manually refine the mask afterwards.

As shown in Table III, although our trained models have shown strong benchmark performance, both models has shown degraded performance on RT 100. This performance degradation is not surprising considering that the viewing angle is significantly different, but it highlights the need on improving the model generalization capability to different viewing angles. To address this, we incorporate RT343 into the training process to adapt the models to vehicle's front view camera.

C. Ablation Study

To evaluate the impact of image augmentation and front view camera image incorporation, we conduct a systematic ablation study in which the augmentation strategy and training dataset are varying factors. The model performance is evaluated on both the RT100 and CRKWH100 testing sets, and the results are concluded in Table IV.

We begin with baseline models trained without augmentation. Specifically, DeepCrack with pre-trained weights is used as Baseline 1, and our reconstructed LECSFormer trained on CrackTree260 without augmentation establishes a performance floor of 0.2923 ODS on RT100.

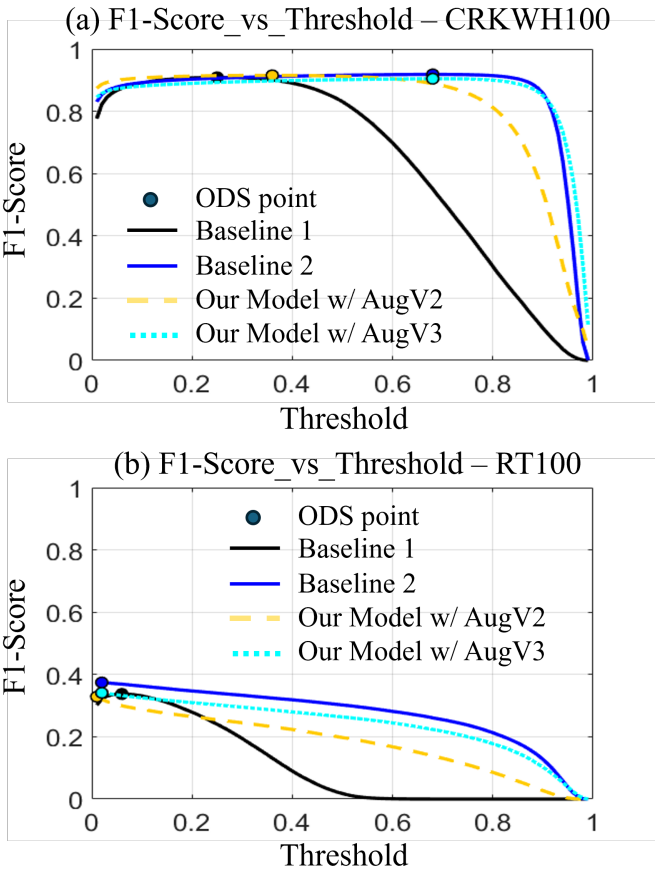


Fig. 5. F1-threshold curves across datasets: (a) CRKWH100 benchmark and (b) RT100 test set comparing five configurations: Baseline 1 (DeepCrack pre-trained), Baseline 2 (LECSFormer trained on CT260 with advanced augmentation), Our Model AugV2 (CT260 with DeepCrack augmentation), and Our Model AugV3 (CT260 with enhanced DeepCrack augmentation). The dots indicate ODS values, and Baseline 2 shows the best robustness across threshold variations on real-time data.

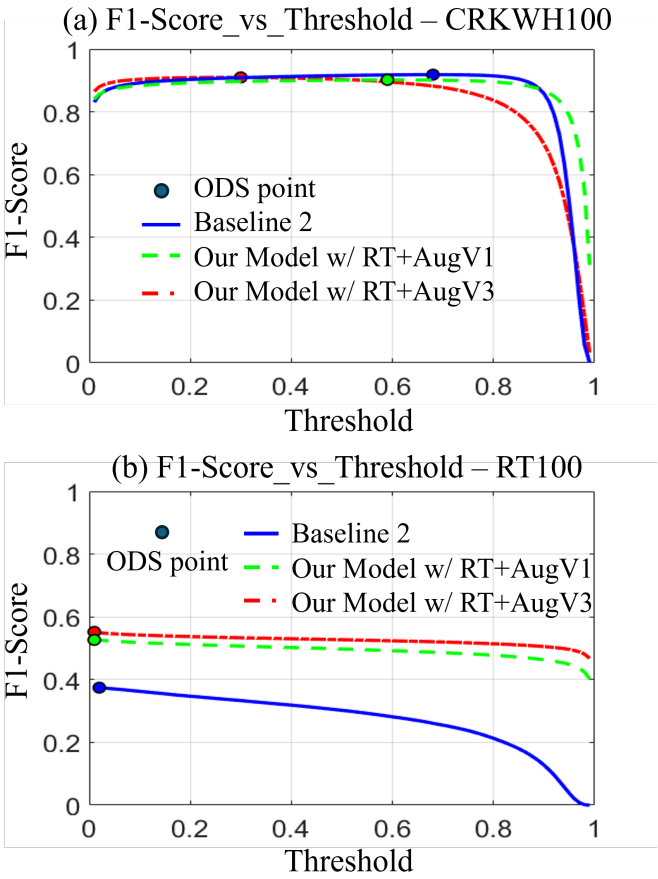


Fig. 6. F1-threshold curves on the (a) CRKWH100 dataset and (b) RT100 test set for models incorporating front view camera image: Baseline 2 (LECSFormer trained on CT260 with advanced augmentation), Our Model w/ RT+AugV1 (trained on CrackTree260 + real-time images with Advanced Augmentation V1), and Our Model w/ RT+AugV3 (trained on CrackTree260 + real-time images with Enhanced DeepCrack Augmentation V3). Integration of domain-specific data with sophisticated augmentation yields significantly higher and more stable performance across thresholds.

TABLE IV
ABLATION STUDY: REAL-TIME PERFORMANCE (ODS F1)

Training Configuration	ODS (RT100)
Baseline Models	
DeepCrack (pre-trained) (**Baseline 1**)	0.3374
LECSFormer (CT260 - no augmentation)	0.2923
Augmentation Effect (CrackTree260)	
DeepCrack augmentation (V2)	0.3287
Enhanced DeepCrack augmentation (V3)	0.3411
Advanced augmentation (V1) (**Baseline 2**)	0.3744
Front View Camera Image Integration	
CT260 + RT343, Aug V1 (**Our model 1**)	0.5267
CT260 + RT343, Aug V3 (**Our model 2**)	**0.5522**

We then evaluate the effect of augmentation by applying three strategies on the CrackTree260 dataset. Our augmentation strategy introduced in Section IV-A2 is denoted as V1, the augmentation strategy originally described in DeepCrack is denoted as V2, an enhanced version of V2 with additional transformation is denoted as V3, respectively. While all augmentation methods improve performance over the no-augmentation cases, V1 and V3 outperform V2 on RT100 with higher ODS scores. Since V3 subsumes the transformations used in V2, we exclude V2 from further analysis.

In the subsequent study, we focus on V1 and V3 for evaluating the impact of front view camera image integration. The model trained on CrackTree260 with V3 augmentation is used as Baseline 2 for comparison. As shown in Table IV, adding the front view camera image into the training set further improved the model performance for both the augmentation strategy V1 and V3. The most significant improvement is achieved when combining front view camera images with the enhanced V3 augmentation, achieving a peak ODS of 0.5522. This is an 89% improvement over the non-augmented LECSFromer trained on CrackTree260 dataset.

In Figs. 5 and 6, we present the F1-score curves of all models listed in Table IV to demonstrate their performance on both the benchmark testing set (CRKWH100) and the

real-time testing set(RT100). Fig. 5 compares the performance between Baseline 1 (DeepCrack with pre-trained weights) and LESCFormer models trained using three augmentation strategies. It shows that models trained with higher augmentation factors (Baseline 2 and V3) outperform the model trained with less augmentation factor (V1) on both two testing sets.
Fig. 6, on the other hand, further compares the model performance between the models trained with real-time dataset incorporation (Our Model w/ RT+ AugV1 and Our Model w/ RT+AugV3) and Baseline 2. It shows a mixed trend: Incorporating real-time training set slightly degrades the model performance below the Baseline 2 on the benchmark testing set (CRKWH100), but significantly improved the performance on RT100. Specifically, F1-score improves from (0.2-0.4) to (0.4-0.6) within threshold range of (0-0.8). This finding underscores the importance of domain-specific data is in closing the generalization gap to different view images.

D. Model Selection

Our study on adapting crack detection models to vehicle front view camera images indicates that the benchmark performance does not directly translate to the real-world deployment. As shown in Table II and Table III, models achieving high accuracy on curated dataset show a significant performance degradation when applied to front view camera images. Furthermore, while image augmentation improves the model robustness, it is not sufficient alone. Although all augmentation strategies enhance performance, their impact remains limited without adding domain-specific data. Performance are improved combining both the augmentation strategy with domain-specific real-world training data; thus, we select the model trained with augmentation V1 (our model 1) and V3 (our model 2) as our final models to be used in the full crack detection pipeline.

V. Full Pipeline Implementation

Building on the CV2X view selection algorithms and crack detection model reported in the previous sections, we are ready to establish a complete infrastructure-guided connectivity-enhanced road crack detection and estimation pipeline. In this section, we first explain the workflow of our full pipeline shown in Fig. 2. To validate our detection results, we used a widely used ground-truth measurement approach with a depth camera, ZED2, and compared our crack length estimates with manual inspection results to present our findings.

A. Full pipeline overview

The complete detection pipeline requires close cooperation between the C-V2X road side unit (RSU) and the software stack that includes C-V2X on-board unit (OBU) on our CAV platform. The software architecture is developed based on ROS2 [35] and Autoware [36]. It contains four carefully designed ROS2 nodes that support real-time detection: the OBU node, the Vehicle node, the Camera node, and the Detection node. Upon receiving a detection request from the RSU, the four nodes collaborate through three sequential

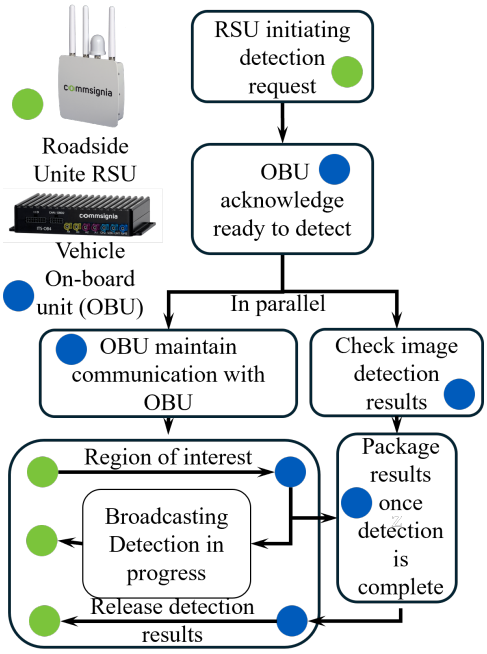


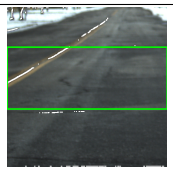
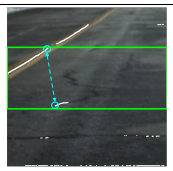
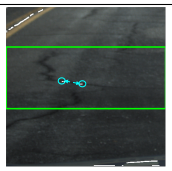
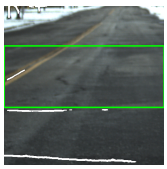
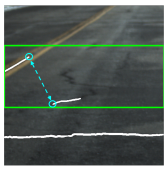
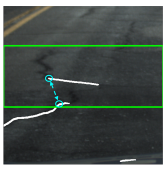
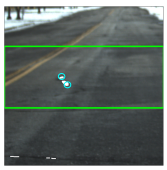
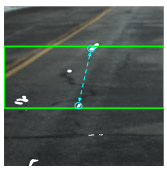
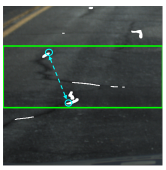
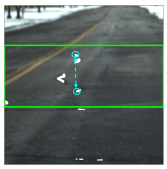
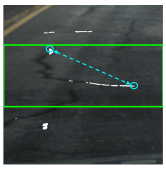
Fig. 7. RSU–OBU communication protocol used for AOI dissemination and feedback exchange with a vehicle-resident crack detection pipeline.

stages to complete the detection task and send the result back to the RSU.

1) Stage **I***: Pre-detection:* Stage **I** assigns detection tasks to the CAV through a specially designed RSU-initiated request–response workflow protocol between the RSU and the OBU, as shown in Fig. 7. The RSU continuously looking for the signals from the OBU running on the CAV. Once the CAV enters the RSU communication range and gets detected, the RSU transmits the crack center coordinates along with relevant metadata. Meanwhile, the OBU node starts sending the received crack coordinate information along with the traffic data, including vehicle information from surrounding CAVs and the status of the ego CAV itself, to the vehicle node. This step enables CAV to navigate to the crack location for detection and send back the result once the detection is done.

2) Stage **II***: In-detection:* In parallel with subscribing to the crack location, the vehicle node also subscribes to all the CAV status from both the OBU node and all the onboard sensor flows. With real-time CAV location along with the crack location, the vehicle node computes the distance between the crack and the front-view camera lens using (1). This distance is further used to decide whether the crack resolution is above threshold for further detection model inference. Once the range satisfies the resolution requirement for the ROI, the image subscription starts. To allow proper synchronization, raw images are logged with timestamp for downstream synchronization. After passing the crack, the images are synced with other information (e.g., GPS) and the cropping windows for images are determined and saved along with each images.

TABLE V
COMPARISON OF CRACK LENGTH ESTIMATION UNDER DIFFERENT RESOLUTIONS. GROUND TRUTH LENGTH = 3.24 M; ZED2 REFERENCE = 3.09 M.

Model	Target Crop	Mask (Low)	Mask (Mid)	Mask (High)	Estimated Length [m] (Low / Mid / High)
Baseline 1					NA / 8.0069 / 0.2464
Baseline 2					NA / 5.7899 / 1.5224
Our Model w/ RT + Aug V1					1.4788 / 8.1916 / **3.3506**
Our Model w/ RT + Aug V3					7.7514 / 0.0865 / **2.9499**

3) Stage **III***: Post-detection:* In this final stage, the detection node accesses the logged data directory and and extracts a 512×512 window from each raw images. These cropped images will be passed on to the detection model trained in Section IV to generate the detected masks. For each mask, the detection node identifies the edge coordinates using the method introduced in Section III-B2 and computes the Euclidean distance between edge points in the second and fourth quadrants as the estimated crack length. The computed length results and the mask images are packaged and transmitted back to the RSU following the protocol in Fig. 7.

B. Ground truth validation and performance evaluation

To validate our pipeline's estimated crack length, we use a ZED 2i stereo camera with its software development kit [37], which provides direct metric depth through stereo triangulation, to compute the crack length result using the same pinhole camera model described in (2). Results in Table. V compares the crack length estimates at different resolutions of the center cropping using four different models. The ground truth corresponds the the red dashed line marked with "Target crack" in the "Target Crop" column. In the Mask columns, the edge corners determined using Algorithm 1 are marked as cyan dots and the corresponding cyan dashed line segments corresponds to the crack length estimation. Crops obtained when the vehicle is far from the crack result in lower effective resolution, as the crack occupies fewer pixels in the image, whereas crops obtained when the vehicle is closer to the crack provide higher effective resolution, with more pixels representing the crack. The results show that models operating at high resolutions consistently outperform those at low resolutions, further emphasizing the importance of the motivation introduced in Section II-B. In addition, our trained model produces length estimates that are closer to those computed using the ZED 2i stereo depth, particularly as the crack-to-camera distance decreases. Moreover, the Baseline 2 model outperformed Baseline 1 in the area of the thin designated crack at high resolution, whereas both our models demonstrate superior detection of the crack's thin and low-contrast sections compared with both baselines.

VI. CONCLUSION

In this paper, we designed an infrastructure-guided communication-enhanced road crack detection pipeline. Leveraging C-V2X communication between the infrastructure and the vehicle, the proposed framework dynamically localizes the region of interest within the vehicle's field of view, streamlines subsequent model inference, and estimates length. We investigated the performance of the state-of-the-art detection models on our carefully prepared real-time testing set (RT100). Our results demonstrate a 89% improvement in the model performance when testing on front-view camera images. In addition, the estimated crack lengths, based on calibrated camera extrinsic parameters, achieve an accuracy of approximately 90% compared to ground truth measurements.

For future work, we will focus on improving the robustness of the proposed platform, particularly by evaluating its performance across diverse terrains and in scenarios involving multiple cracks within a single region of interest (ROI). Furthermore, we aim to develop an adaptive auto-focus mechanism that better localizes the region of interest to enhance dynamic cropping and further improve detection performance, integrating vehicle motion control [38].

ACKNOWLEDGMENT

The authors acknowledge the financial support provided for this study by the Transportation Infrastructure Precast Innovation Center (TRANS-IPIC) through the University Transportation Center program of the US Department of Transportation, Office of the Assistant Secretary for Research and Technology (OST-R) under Grant No. 69A3552348333.

REFERENCES

[1] American Society of Civil Engineers. A comprehensive assessment of america's infrastructure. *A Comprehensive Assessment of America's Infrastructure, ASCE, ASCE*, pages 1–32, 2025.

[2] X. Q. Kong, Z. H. Li, Y. L. Zhang, and S. Das. Bridge deck deterioration: Reasons and patterns. *Transportation Research Record*, 2676(7):570–584, 2022.

[3] Chan Yang, Peng Lou, and Hani Nassif. Correlation of bridge deck deterioration with truckload spectra based on nbi condition rating and weigh-in-motion data. Report, United States. Department of Transportation. Federal Highway Administration.

[4] A. Ibrahim, S. Abdelkhalek, T. Zayed, A. H. Qureshi, and E. M. Abdelkader. A comprehensive review of the key deterioration factors of concrete bridge decks. *Buildings*, 14(11), 2024.

[5] Abdelhady Omar. Condition monitoring of reinforced concrete bridge decks: Current practices and future perspectives. *Current Trends in Civil & Structural Engineering*, 8(4), 2022.

[6] R. Tawadrous, G. Morcous, and M. Maguire. Performance evaluation of a new precast concrete bridge deck system. *Journal of Bridge Engineering*, 24(6), 2019.

[7] E. Shahrokhinasab and D. Garber. Long-term performance of full-depth precast concrete (fdpc) deck panels. *Engineering Structures*, 244, 2021.

[8] Michael; McDonagh, Andrew; Foden, and Alexandra Beyer. State-of-the-practice report— partial-depth precast concrete deck panels. Report, United States. Federal Highway Administration. Office of Bridges and Structures; United States. Department of Transportation. Federal Highway Administration. Office of Infrastructure, 2022.

[9] American Association of State Highway and Transportation Officials. *AASHTO guide manual for bridge element inspection*. Guide manual for bridge element inspection. American Association of State Highway and Transportation Officials, Washington, DC, 1st edition, 2011.

[10] The evolution of drones part 3: Building highways in the sky. Report, American Association of State Highway and Transportation Officials, 2019.

[11] Alynix. Drone-driven infrared bridge deck corridor scans. https://alynix.com/decker/. Accessed: 2026-03-25.

[12] NEXCO WEST USA. Deck Top Scanning System (DTSS). https://www.w-nexco-usa.com/documents/technology/DTSS.pdf. Accessed: 2026-01-04.

[13] U.S. Department of Transportation. USDOT releases national deployment plan for vehicle-to-everything (V2X) technologies to reduce death and serious injuries on america's roadways. link, 2024. Press release, Aug. 16, 2024.

[14] KR Kirschke and SA Velinsky. Histogram-based approach for automated pavement-crack sensing. *Journal of Transportation Engineering*, 118(5):700–710, 1992.

[15] Ikhlas Abdel-Qader, Osama Abudayyeh, and Michael E Kelly. Analysis of edge-detection techniques for crack identification in bridges. *Journal of computing in civil engineering*, 17(4):255–263, 2003.

[16] Albert Ayenu-Prah and Nii Attoh-Okine. Evaluating pavement cracks with bidimensional empirical mode decomposition. *EURASIP Journal on Advances in Signal Processing*, 2008(1):861701, 2008.

[17] Qin Zou, Yu Cao, Qingquan Li, Qijie Mao, and Song Wang. Cracktree: Automatic crack detection from pavement images. *Pattern Recognition Letters*, 33(3):227–238, 2012.

[18] Qin Zou, Zheng Zhang, Qingquan Li, Xianbiao Qi, Qian Wang, and Song Wang. Deepcrack: Learning hierarchical convolutional features for crack detection. *IEEE Transactions on Image Processing*, 28(3):1498–1512, 2019.

[19] Junzhou Chen, Nan Zhao, Ronghui Zhang, Long Chen, Kai Huang, and Zhijun Qiu. Refined crack detection via lecsformer for autonomous road inspection vehicles. *IEEE Transactions on Intelligent Vehicles*, 8(3):2049–2061, 2022.

[20] Haitao Li, Tao Peng, Ningguo Qiao, Zhiwei Guan, Xinyun Feng, Peng Guo, Tingting Duan, and Jinfeng Gong. Cracktinynet: A novel deep learning model specifically designed for superior performance in tiny road surface crack detection. *IET Intelligent Transport Systems*, 18(12):2693–2712, 2024.

[21] Yifan Pan, Xianfeng Zhang, Guido Cervone, and Liping Yang. Detection of asphalt pavement potholes and cracks based on the unmanned aerial vehicle multispectral imagery. *IEEE Journal of Selected Topics in Applied Earth Observations and Remote Sensing*, 11(10):3701–3712, 2018.

[22] Xinbao Chen, Chang Liu, Long Chen, Xiaodong Zhu, Yaohui Zhang, and Chenxi Wang. A pavement crack detection and evaluation framework for a UAV inspection system based on deep learning. *Applied Sciences*, 14(3):1157, 2024.

[23] Zhengfei Song, Nachuan Ma, Zhuoyan Zhang, Ming-Ju Lee, Sergey Vityazev, Alexander Dvorkovich, Rui Fan, et al. Robust and real-time road crack detection through collaborative dual-branch learning on robotic sensing platform. In *2025 IEEE International Conference on Real-time Computing and Robotics (RCAR)*, pages 55–60. IEEE, 2025.

[24] Sike Wang, Qiao Dong, Xueqin Chen, Zepeng Chu, Ruiqi Li, Jing Hu, and Xingyu Gu. Measurement of asphalt pavement crack length using yolo v5-bifpn. *Journal of Infrastructure Systems*, 30(2):04024005, 2024.

[25] Sukhad Anand, Saksham Gupta, Vaibhav Darbari, and Shivam Kohli. Crack-pot: Autonomous road crack and pothole detection. In *2018 digital image computing: techniques and applications (DICTA)*, pages 1–6. IEEE, 2018.

[26] Felix Kortmann, Pascal Fassmeyer, Burkhardt Funk, and Paul Drews. Watch out, pothole! featuring road damage detection in an end-to-end system for autonomous driving. *Data & Knowledge Engineering*, 142:102091, 2022.

[27] Nexteer Automotive. Road surface detection. https://www.nexteer.com/software/road-surface-detection/. Accessed: 2026-01-04.

[28] Just Auto Magazine. Connected mercedes cars to transmit road condition data in sweden. https://justauto.nridigital.com/just_auto_magazine_sep23/connected_mercedes_cars_to_transmit_road_condition_data_in_sweden. Accessed: 2026-03-25.

[29] E. Ranyal, A. Sadhu, and K. Jain. Road condition monitoring using smart sensing and artificial intelligence: A review. *Sensors (Basel)*, 22(8), 2022.

[30] Nachuan Ma, Qiang Hu, Zhengfei Song, Sicen Guo, and Rui Fan. Self-derived multi-modal knowledge distillation for real-time road crack detection. *IEEE Sensors Journal*, 2025.

[31] Qualcomm Technologies Inc. Cellular-v2x technology overview, 2019.

[32] OpenCV Developers. Camera calibration. https://docs.opencv.org/4.x/dc/dbb/tutorial_py_calibration.html, 2024. Accessed: 2026-01-05.

[33] Ultralytics. YOLOv8 Documentation. https://docs.ultralytics.com/models/yolov8/, 2023.

[34] Nachuan Ma, Zhengfei Song, Qiang Hu, Chuang-Wei Liu, Yu Han, Yanting Zhang, Rui Fan, and Lihua Xie. Vehicular road crack detection with deep learning: A new online benchmark for comprehensive evaluation of existing algorithms. *arXiv preprint arXiv:2503.18082*, 2025.

[35] Open Robotics. ROS 2: Robot Operating System 2. https://docs.ros.org/en/foxy/index.html, 2022. Accessed: 2026-03-22.

[36] Autoware Foundation. Autoware: Open-Source Software for Autonomous Driving. https://github.com/autowarefoundation/autoware, 2023. Accessed: 2026-03-22.

[37] StereoLabs. ZED SDK. https://www.stereolabs.com/developers, 2026. Accessed: Jan. 4, 2026.

[38] Haosong Xiao and Chaozhe R He. Safe and efficient data-driven connected cruise control. *IFAC-PapersOnLine*, 59(30):749–754, 2025.