



From instance segmentation to physical quantification: High-resolution UAV-based dataset for façade defect assessment

Benyun Zhao¹, Jihan Zhang^{1,*}, Yijun Huang¹, Xi Chen^{1,*}, Ben M. Chen¹

Department of Mechanical and Automation Engineering, The Chinese University of Hong Kong, Shatin District, N.T., Hong Kong

ARTICLE INFO

Dataset link: <https://github.com/BenyunZhao/CUBIT-InSeg>

Keywords:

Building inspection
Unmanned aerial vehicle
Deep learning
Instance segmentation
Defect segmentation
Defect assessment
Defect dataset

ABSTRACT

High-rise building façade inspection is essential but hazardous and labor-intensive. Although UAV imaging and defect segmentation methods can recognize defects, most studies stop at 2D masks and lack the scale and instance structure needed for grade quantification. Public façade datasets remain scarce, often low-resolution, crack-only, and annotated for semantic segmentation. This paper introduces CUBIT-InSeg, a high-resolution UAV building façade defect instance segmentation dataset. It contains 6996 images at 4800×3200 with 62,187 annotated instances of two safety-critical defects, cracks and spalling, in realistic multi-instance scenes. Over 80 models were benchmarked under a unified protocol, and transferability was assessed via zero-shot testing on two cross-domain datasets. A deployment workflow was further demonstrated that registers defects to an as-is digital twin and converts segmentation masks into metric measures (e.g., crack width/length, spalling area) for standard-aligned severity evaluation. In real-world validation, the workflow based on CUBIT-InSeg dataset enables reproducible research toward scalable, quantitative façade condition assessment.

1. Introduction

Civil infrastructure is vulnerable to damage induced by environmental exposure, external loads, material deterioration, and construction deficiencies. For high-rise buildings, façade defects such as cracks and concrete spallings not only degrade durability but may also pose safety hazards, motivating periodic inspections for condition assessment and maintenance planning. In practice, close-range visual inspection and non-destructive testing (NDT) devices (e.g., optical imaging [1], laser scanning [2], impact echo [3], and ground-penetrating radar [4]) are commonly used for defect investigation. However, human-centric inspection is subjective, labor-intensive, and potentially hazardous, especially for high-rise façades and large inspection areas. To improve safety and efficiency, robotic platforms such as unmanned aerial vehicles (UAVs) have been increasingly adopted to facilitate façade data collection and automated defect analysis [5]. Meanwhile, deep learning has enabled rapid progress in visual recognition [6], outperforming many traditional feature-based pipelines [7], and has been increasingly adopted in the architecture, engineering, and construction (AEC) domain [8]. For façade inspection, pixel-level understanding is crucial since defects are often thin, irregular, spatially distributed, and strongly multi-scale, while multiple defects can co-exist in the same view. In this context, instance segmentation is particularly suitable due to providing both instance-level localization and pixel-accurate masks, which not only separate multiple defect instances but also directly support

downstream quantification, instance-wise engineering statistics, and interpretable severity evaluation for maintenance decision-making [9].

Despite this progress, deep learning models are notoriously data-hungry and require large, high-quality, domain-specific datasets. Models trained on generic datasets such as MS COCO [10] do not transfer reliably to building façade defects due to substantial differences in viewpoint, surface texture, scale distribution, and background clutter, while dense pixel-level annotations are expensive in real inspection environments. More importantly, from the perspective of practical façade inspection, the ultimate objective is not merely visual recognition, but severity grading for condition evaluation and maintenance prioritization, which in turn requires a reliable defect modeling pipeline based on physically interpretable defect attributes. Public defect datasets are therefore scarce and predominantly focus on pavement scenarios; many are crack-only, low-resolution, and annotated for binary semantic segmentation rather than instance segmentation [11]. This limitation is particularly restrictive for our task, because semantic labels alone are insufficient when downstream engineering analysis requires defect-wise localization, physical quantification, and aggregation at the façade-component level. In particular, UAV-based façade inspection must maintain a non-negligible standoff distance from the wall for flight safety and coverage; at such distances, low-resolution imagery often fails to resolve fine-grained defects (e.g., tiny cracks),

* Corresponding author.

E-mail addresses: jhzhang@mae.cuhk.edu.hk (J. Zhang), xichen002@cuhk.edu.hk (X. Chen).

<https://doi.org/10.1016/j.autcon.2026.106980>

Received 21 January 2026; Received in revised form 14 April 2026; Accepted 20 April 2026

Available online 24 April 2026

0926-5805/© 2026 The Authors. Published by Elsevier B.V. This is an open access article under the CC BY-NC-ND license (<http://creativecommons.org/licenses/by-nc-nd/4.0/>).

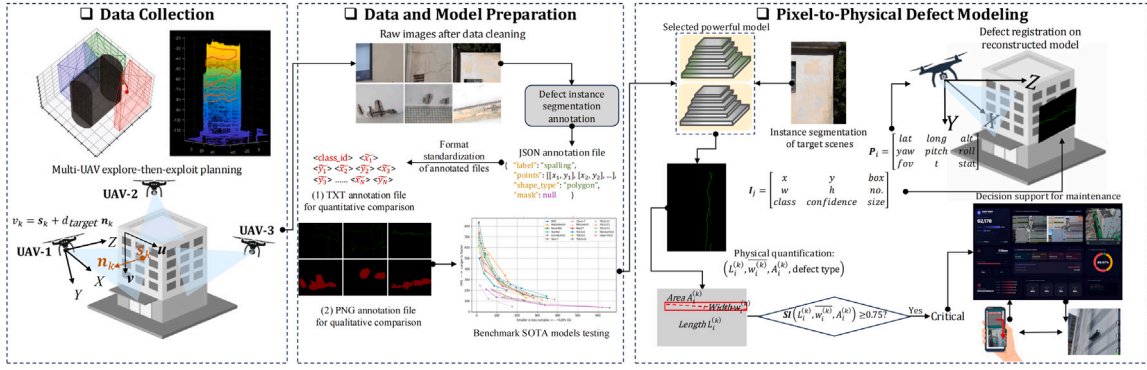


Fig. 1. Overall pipeline of the pixel-to-physical modeling for infrastructure defects.

Table 1
CUBIT-InSeg dataset card.

Attribute	Description
Dataset Name	CUBIT-InSeg dataset, where CUBIT stands for <u>C</u> UHK <u>B</u> uilding <u>I</u> nformation <u>T</u> echnology and InSeg stands for <u>I</u> nstance <u>S</u> egmentation
Modality	RGB images
Sensor Type	High-resolution RGB forward-looking cameras mounted on UAVs
Resolution	4800 × 3200
Target Type	Crack, Spalling
Dataset Scale	6996 images with 62,187 instances
Splits	Train (5596)/val (699)/test (701)
Annotation Type	Bounding Box (Det) + Mask (Seg)

rendering many existing datasets impractical for realistic UAV façade defect segmentation [12]. Moreover, translating pixel-level predictions into metric, standard-aligned severity indicators remains insufficiently standardized in existing practice, partly because most public datasets provide RGB images and visual labels only, without controlled acquisition geometry or physical scale information needed for reproducible quantification and condition grading [13].

To address the gap, we present CUBIT-InSeg, a UAV-collected high-resolution building façade defect dataset with *instance-level* annotations tailored for high-rise inspection and downstream physical quantification. The starting point of this paper is therefore not simply to release a new dataset, but to establish the necessary data foundation for a severity-grading-oriented technical route. During the development of our defect physical modeling workflow, we found that existing public datasets could not adequately support this objective due to their limited resolution, annotation granularity, scene realism, and lack of compatibility with controlled UAV inspection geometry. CUBIT-InSeg contains 6996 high-resolution images at 4800 × 3200 acquired by UAVs, with 62,187 annotated defect instances. The dataset covers two safety-relevant defect types, cracks and spallings, and includes realistic multi-instance inspection scenes with complex textures, illumination changes, and aerial viewpoints. Moreover, the dataset is primarily collected under an approximately equal-distance imaging strategy, and is compatible with multi-UAV collaborative acquisition in practice, better reflecting the operational geometry of high-rise façade inspection. In this sense, CUBIT-InSeg is characterized not merely by its dataset scale, but by its deliberate design as an engineering bridge from high-resolution façade defect instance segmentation to downstream physical quantification, defect modeling, and DT-oriented severity assessment.

Beyond releasing the dataset, we establish reproducible baselines through systematic benchmarking to provide a unified reference for model comparison and to examine the usability of CUBIT-InSeg as a benchmark resource. We evaluate more than 80 representative state-of-the-art (SOTA) models under a unified training and evaluation

protocol, spanning (1) End-to-end frameworks covering both CNN-based and transformer-based paradigms, and (2) Single-stage, real-time, deployment-oriented YOLO-style variants. To assess robustness and external validity, we further conduct cross-domain zero-shot evaluation by directly testing models trained on CUBIT-InSeg on two public infrastructure crack datasets without target-domain fine-tuning (Highway-Crack [14] and Crack-Seg [15]).

Finally, while many studies of defect recognition only report in-the-lab recognition metrics (e.g., AP/IoU), practical inspection workflows require more than visual recognition alone. Accordingly, we connect instance-level predictions to a downstream deployment-oriented, digital twin (DT) quantification workflow. Under the equal-distance acquisition assumption, pixel masks are converted into physically meaningful measures such as crack width/length and spalling area, while instance-level localization (bounding boxes) supports defect registration and aggregation at the façade component level. These component-level indicators enable standard-aligned severity grading consistent with BS ISO 15686-7:2017 [16] and local façade inspection practice in Hong Kong [17,18]. Based on the quantified crack and spalling metrics, we derive component-level severity ranks and store the results in the DT to support reporting, visualization, and maintenance decision-making. This establishes a closed-loop workflow from visual sensing and instance segmentation to metric computation, component-level assessment, and DT-enabled decision support, as illustrated in Fig. 1. To further demonstrate the practical applicability of this pipeline, we select YOLOv9 [19] as the best-performing model based on comprehensive results across CUBIT-InSeg and two cross-domain test sets, and present a real high-rise façade inspection scenario with DT-oriented registration, metric quantification, and severity rating.

The main contributions are summarized as follows:

- **A high-resolution façade defect instance segmentation dataset.** We curate and annotate 6996 high-resolution façade images at 4800 × 3200 with 62,187 defect instances, covering safety-relevant defect types (crack and spalling) and realistic multi-instance inspection scenes. A compact dataset card is provided in Table 1 to give a quick overview of the dataset characteristics.
- **Systematic benchmarking of more than 80 models to validate dataset usability.** We conduct unified training and evaluation on more than 80 representative models, including end-to-end instance segmentation families (covering both CNN-based and transformer-based paradigms) and a broad set of single-stage YOLO-style variants, providing reproducible baselines for the community.
- **Cross-domain zero-shot evaluation to assess transferability.** We directly test models trained on CUBIT-InSeg on two public cross-domain datasets in a zero-shot manner (i.e., without target-domain fine-tuning), which provides a practical protocol for examining transfer performance across related inspection scenarios.

- **DT-oriented defect registration, physical quantification, and severity grading.** We select the best-performing model for real-world high-rise façade inspection, and demonstrate a complete pipeline from instance-level predictions to defect registration (via instance footprints and bounding boxes), metric quantification (via pixel masks), and standard-aligned severity grading on a DT.

Overall, these contributions form a coherent pipeline rather than a set of isolated components. The practical need for severity grading motivates the construction of a high-resolution, UAV-oriented instance segmentation dataset as the necessary data foundation. Based on this foundation, the large-scale benchmark and cross-domain evaluation establish reproducible baselines and verify the usability of the dataset for method development. The subsequent DT-oriented physical modeling workflow then demonstrates how instance-level segmentation outputs can be further translated into engineering-relevant defect descriptors and severity indicators. From the perspective of reproducibility, we consider the dataset to be the primary entry point for the community, while the benchmark protocol and the DT-oriented workflow provide reference pathways for follow-up studies on model evaluation and deployment-oriented defect assessment. The remainder of this paper is organized as follows. Section 2 reviews the related works of our research. Section 3 introduces the methodology of dataset establishment. Section 4 presents the physical quantification and DT integration workflow. Section 5 reports the benchmark and cross-domain evaluation results. Section 6 presents the real-world DT-oriented façade inspection case. Section 7 makes a conclusion, and discusses the limitations.

2. Related work

We reviews the related research that is relevant to this paper from three perspectives. Section 2.1 surveys public infrastructure defect datasets, with particular attention to collection scenarios and annotation granularity, in order to clarify the remaining gap in UAV-based façade instance segmentation. Section 2.2 then summarizes representative defect segmentation methods, spanning semantic segmentation, end-to-end instance segmentation, and deployment-oriented single-stage frameworks. Section 2.3 further reviews recent efforts on defect quantification and DT integration for UAV-based inspection, thereby linking visual defect recognition with downstream physical assessment and management-oriented deployment.

2.1. Infrastructure defect datasets

Existing infrastructure defect datasets can be understood from two complementary perspectives: (1) collection scenario and (2) annotation granularity/task type. From the *scenario* perspective, public datasets are dominated by pavement surfaces (such as GAPS series [20,21] and its variants GAPS-10m [22], Crack500 [23] and EdmCrack600 [24]), followed by the deck of bridge scenarios (Xu [25] and Philipp [26]), while datasets targeting buildings [27] are the scarcest. More importantly, a substantial domain gap often exists across infrastructure types: even for the same defect category (e.g., cracks), the foreground morphology (width, orientation, and scale distribution) and the background appearance (material textures, coatings, and stains) can differ markedly with the asset type and viewing geometry, making cross-scenario reuse of data and models non-trivial [11]. Many pavement crack datasets [22,24] are captured under near-planar viewpoints with relatively homogeneous textures and predominantly focus on crack-type defects, which inherently limits their transferability to façade inspection where textures are more complex, viewpoints are oblique, illumination varies significantly, and defect scales span a wider range. Even among unmanned-system captured datasets, Highway-Crack [14] is one of the few UAV-acquired datasets; however, it remains constrained to post-earthquake highway surfaces and does not reflect the operational geometry and visual characteristics of high-rise façade

inspection. Public datasets for building façades are even more limited: Crack-Seg [15] contains partial building scenes but lacks UAV-faithful viewpoints and sufficient resolution, while UAV75 [28] provides UAV imagery but includes only a small number of low-resolution samples, making it inadequate for training and benchmarking modern deep segmentation models.

From the annotation perspective, a clear granularity gap also exists. Early resources often adopt image-level *classification* annotations for defect presence or coarse categories, like CCIC [29], GAPS series [20, 21], Xu et al. [25] and Philipp et al. [26] proposed datasets. Later datasets introduce box-level *detection* labels for coarse localization, like RDD series [30,31] and Zhao et al. [27] proposed datasets. More commonly, datasets provide *semantic segmentation* mask, typically binary (one defect type vs. background), especially for cracks, such as GAPS-10m [22], Crack500 [23], Highway-Crack [14], Crack-Seg [15], EdmCrack600 [24] and UAV75 [28]. While useful for pixel-level delineation, semantic masks do not separate multiple instances of the same defect type and are less suitable for instance-wise statistics, tracking, and component-level registration. In contrast, instance segmentation datasets that jointly support object-level localization and pixel-accurate boundaries are much rarer, particularly under UAV viewpoints and for building façades.

To further expose the practical gap in unmanned system-captured datasets for segmentation-oriented tasks, Table 2 compares recent defect segmentation datasets acquired by unmanned platforms (UAV/UGV). This comparison shows that UAV-acquired datasets are scarce, with only two such datasets identified, and only one of them targets building scenarios. Moreover, these UAV datasets provide pixel-level semantic masks rather than instance-level annotations. Quantitatively, compared with UAV75 dataset, which is the only prior UAV building dataset listed in Table 2, CUBIT-InSeg increases the image resolution from 512×512 to 4800×3200 and the number of images from 75 to 6996, while additionally providing 62,187 instance-level annotations. Together with the commonly observed limitations in resolution, defect density, which often corresponds to single-defect images, and defect-type diversity, existing resources remain insufficient for realistic high-rise façade inspection and downstream engineering assessment. Based on Table 2, several representative datasets are further compared qualitatively in Fig. 2, which more intuitively illustrates that these existing datasets not only focus on a single defect type, namely cracks, but also contain annotation errors, highlighted by yellow ellipses.

Overall, these observations call for a public benchmark that is simultaneously *high-resolution*, *UAV-faithful*, *façade-centric*, *multi-type*, and *instance-level*, enabling reproducible training and evaluation, cross-domain validation, and subsequent workflows such as defect registration, metric quantification, and standard-aligned severity grading. CUBIT-InSeg is constructed to fill this gap.

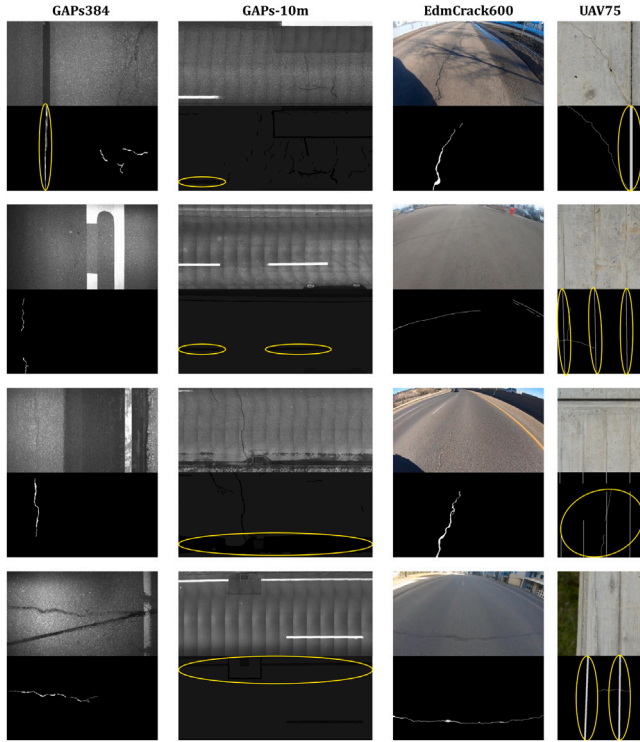
2.2. Defect segmentation methods

Defect segmentation methods can be broadly categorized into semantic segmentation and instance segmentation. Semantic segmentation models (e.g., U-Net and DeepLab families) have been widely used for crack/spalling region extraction in civil infrastructure [32,33]. These methods are effective for pixel-wise defect region delineation and have played an important role in infrastructure inspection, especially for crack-focused studies. Recent works have further explored task-specific designs for thin and elongated crack patterns under complex backgrounds. For example, DYOLO [34] incorporates dynamic snake convolution into a YOLOv8-seg framework to better adapt to tortuous crack morphology and preserve fine structural continuity in challenging tunnel environments. Although developed for tunnel lining cracks, this line of work is highly relevant to façade inspection because façade cracks also exhibit slender geometry, low contrast, and strong background interference. At the same time, façade inspection is not limited to cracks alone, but also involves other defect categories such

Table 2

Comparison between other unmanned system-captured defects segmentation dataset with CUBIT-InSeg dataset.

Dataset	Image volume	Resolution	Data collection platform	Defect type	Infrastructure type	Material type	Task type
GAPs384 [23]	384	1920 × 1080	Ground Vehicle	Crack	Pavement	Asphalt	Pixel Level
GAPs-10m [22]	20	5030 × 11505	Ground Vehicle	Crack	Pavement	Asphalt	Pixel Level
EdmCrack600 [24]	600	1920 × 1080	Ground Vehicle	Crack	Pavement	Asphalt	Pixel Level
Highway-Crack [14]	4118	512 × 512	Unmanned Aerial Vehicle	Crack	Pavement	Asphalt	Pixel Level
Crack-Seg [15]	4,029	416 × 416	Ground Vehicle	Crack	Building Pavement	Concrete Asphalt	Pixel Level
UAV75 [28]	75	512 × 512	Unmanned Aerial Vehicle	Crack	Building	Concrete	Pixel Level
CUBIT-InSeg (Ours)	6,996 62,187 instances	4800 × 3200	Unmanned Aerial Vehicle	Crack Spalling	Building	Concrete	Instance Level

**Fig. 2.** Samples of existing defect segmentation datasets collected by unmanned systems.

as spalling, efflorescence, and moisture-related deterioration, which makes broader defect representation necessary.

Despite their usefulness, semantic segmentation methods typically cannot separate multiple defect instances in the same image, which limits instance-wise counting, tracking, registration, and quantitative reporting. This limitation is particularly relevant in UAV-based façade inspection, where multiple defects of different categories may co-exist within a single high-resolution image. Instance segmentation provides both instance-level localization and pixel-accurate masks, making it more suitable for engineering-oriented defect representation and downstream quantification. Mainstream approaches include (1) end-to-end two-stage or unified frameworks (e.g., Mask R-CNN [35] and its extensions) that often achieve strong accuracy but with higher computational cost; (2) transformer-based instance/panoptic segmentation frameworks that leverage global context for robustness under complex textures, such as Deformable-DETR [36] and MaskFormer [37]; and (3) deployment-oriented single-stage instance segmentation, often represented by YOLO-style variants, which emphasize efficiency and real-time capability, such as YOLOv8 [38] and Hyper-YOLO [39]. Multi-modal large vision model-driven segmentation (e.g., SAM [40]) has also

attracted attention, but reliable fine-defect delineation and consistent instance outputs in engineering settings still depend on domain-specific data and evaluation benchmarks [41]. Given the rapidly evolving model space and the strong dependence on data properties, we conduct unified benchmarking on more than 80 representative models to establish reproducible baselines and support practical model selection.

2.3. Defect quantification and DT integration for UAV-based inspection

With the widespread adoption of UAVs in façade and infrastructure inspection, vision-based defect detection has become a de facto solution in both research and practice. However, many current UAV deployments still rely on largely manual or experience-driven acquisition strategies, where image quality, viewing geometry, and data consistency are difficult to control [42]. Such practices limit not only the achievable detection accuracy, but also the reproducibility, scalability, and traceability of inspection results across buildings and over time [43]. As a result, there is an increasing need to couple UAV-based visual inspection with data- and management-oriented requirements, including controlled image quality, consistent acquisition geometry, and reliable data provenance, to support systematic inspection workflows rather than isolated visual surveys [44].

Alongside data acquisition, a more critical challenge lies in how high-accuracy detection and segmentation results are utilized for inspection management and engineering decision-making. While advances in learning-based perception have gradually shifted research attention from defect detection toward physically meaningful defect modeling and condition assessment [45], many existing studies still emphasize visual outputs or geometric reconstruction without delivering actionable quantitative indicators. Earlier works focused primarily on identifying defect regions in images, such as road damage detection [46] and building defect recognition [47], whereas more recent efforts aim to recover geometric and metric attributes to support engineering evaluation. Two complementary directions have emerged: integrating multi-view photogrammetry or depth-enhanced imaging to reconstruct cracks or damaged regions in 3D for shape-aware [48] or volume-aware assessment [49], and leveraging point clouds with graph-based learning to build defect representations enriched with geometric and semantic attributes [50]. In parallel, aligning UAV observations with DT representations has been explored as a practical route toward component-level localization and life-cycle asset management [13].

Despite this progress, real-world façade inspection remains constrained by the lack of reproducible physical scale and standard-aligned severity indicators in public datasets and evaluation protocols [51]. In high-rise inspection scenarios, UAVs must operate at a safe standoff distance; without controlled acquisition geometry or explicit scale assumptions, pixel-level predictions cannot be reliably converted into crack width, length, or spalling area, nor can they be consistently registered and aggregated at the component level for assessment [52]. CUBIT-InSeg is therefore collected under an approximately equal-distance acquisition strategy to facilitate metric conversion from instance-level predictions, while enabling defect registration and DT-oriented aggregation for quantitative façade condition assessment.

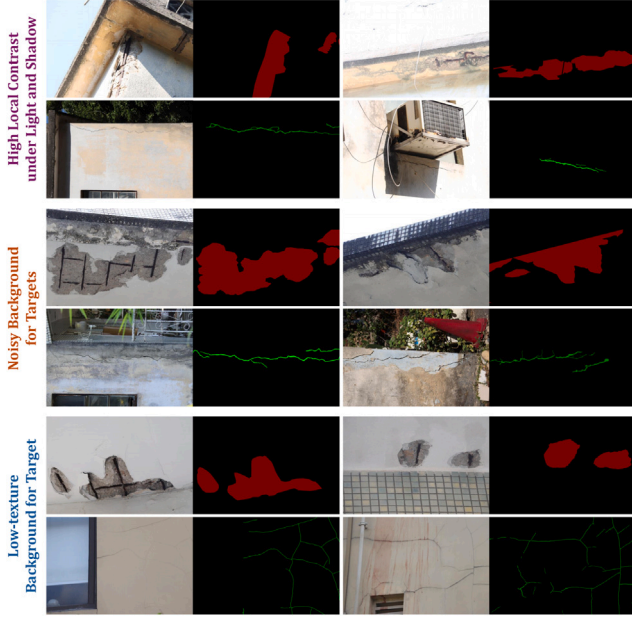


Fig. 3. Samples of the proposed CUBIT-InSeg dataset.

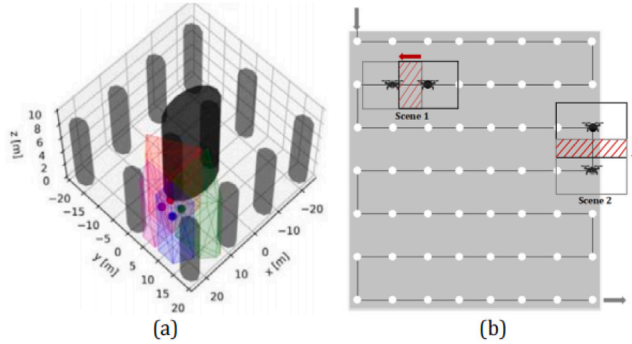


Fig. 4. (a) Multi-UAV data collection for target infrastructure; (b) Each UAV flight strategy for façade.

3. Dataset establishment and analysis

In this paper, we develop an equal-distance UAV imaging framework for the establishment of a high-quality façade defect instance segmentation dataset, as illustrated in Fig. 1. The sample images of CUBIT-InSeg are shown in Fig. 3. The central idea is to capture façade images from a prescribed, nearly constant standoff distance to the building surface, so as to maintain a stable image scale across viewpoints and façades. This acquisition strategy is important for improving annotation consistency, defect visibility, and cross-image comparability in UAV-based façade inspection. To achieve this, our framework extends recent explore-then-exploit multi-UAV coverage schemes (shown in Fig. 4) for infrastructure inspection [12] and integrates them with equal-distance viewpoint generation and façade-oriented inspection planning, forming the basis for reproducible dataset collection and subsequent analysis.

3.1. Path planning for data collection

We adopt an explore-then-exploit paradigm for multi-UAV façade inspection, following recent hierarchical coverage frameworks. During the exploration stage, each UAV performs depth-visual SLAM to estimate its six-degree-of-freedom pose and incrementally reconstruct

the façade surface and surrounding obstacles. The environment is discretized into a 3D occupancy grid, where each voxel stores occupancy probability, distance to the nearest obstacle, and a reconstructability score reflecting multi-view observability and safety margins. This yields a density map $R(\mathbf{p})$ that guides coverage toward regions with high reconstruction potential.

Let $W \subset \mathbb{R}^3$ denote the workspace and $P(t) = [\mathbf{p}_1(t), \dots, \mathbf{p}_n(t)]$ the UAV positions. To coordinate multiple UAVs, the workspace is partitioned into Voronoi cells $V_i(t)$, and a density-weighted coverage cost

$$H(P) = \sum_{i=1}^n \int_{V_i} \|\mathbf{p} - \mathbf{p}_i\|^2 R(\mathbf{p}) d\mathbf{p} \quad (1)$$

is minimized to achieve a centroidal Voronoi deployment. Collision avoidance is ensured by intersecting the Voronoi cells with hyperplane-based safe convex corridors, so that UAV motions remain within collision-free regions.

Once exploration converges and a sufficiently accurate façade surface model is obtained, the exploitation stage generates equal-distance viewpoints for high-quality imaging. For each reconstructed surface point $\mathbf{s}_k$ with outward normal $\mathbf{n}_k$, a desired camera position is defined as

$$\mathbf{v}_k = \mathbf{s}_k + d_{\text{target}} \mathbf{n}_k, \quad (2)$$

where d_{target} is the nominal standoff distance selected based on the required measurement resolution, camera hardware parameters (focal length f and pixel pitch δ_p), and flight-safety constraints. Viewpoints are further filtered to satisfy overlap, viewing-angle, and collision-free requirements, ensuring a reproducible imaging strategy.

The viewpoints assigned to each UAV are further partitioned using capacity-constrained Voronoi tessellation to account for limited endurance. For each subregion, a traveling salesman problem is formulated with edge weights given by collision-free path lengths, producing short and feasible inspection routes. These routes are finally converted into time-parameterized flight plans (position, yaw, and camera trigger events) compatible with UAV autopilots, enabling automatic and DT-ready equal-distance façade imaging.

3.2. Data cleaning and annotation

After data collection, we perform data cleaning to remove unnecessary or visually unreliable samples, which not only reduces annotation workload but also improves the overall dataset quality. In total, we collected over 8000 raw images. We removed images that contain no defect instances, as well as samples with severe motion blur, defocus, overexposure, underexposure, or other visual conditions under which defect boundaries cannot be reliably judged by human inspection. After cleaning, 6996 images were retained for annotation.

We then conduct instance-level defect annotation using the widely-used open-source annotation tool LabelMe [53], where each defect object is delineated by an independent polygon (i.e., one polygon per instance). The upper part of Fig. 5 illustrates the LabelMe user interface. The left panel displays the image, where annotators click sequential points to form a polygon that tightly encloses a target defect instance. The right panel provides the customized defect category list, the instance list, and the image list to be annotated. During annotation, we follow two strict principles: (1) each defect instance must be fully contained within its corresponding polygon, and the polygon boundary should fit the defect contour as closely as possible; (2) every defect instance that can be reliably identified and delineated in an image should be annotated without omission.

To ensure high-quality and consistent labels, our annotation workflow consists of three stages. First, three student annotators carry out the initial labeling of the images. Second, they cross-check each other's annotations to identify samples with ambiguous or questionable labels. Third, the remaining ambiguities are resolved by the first author

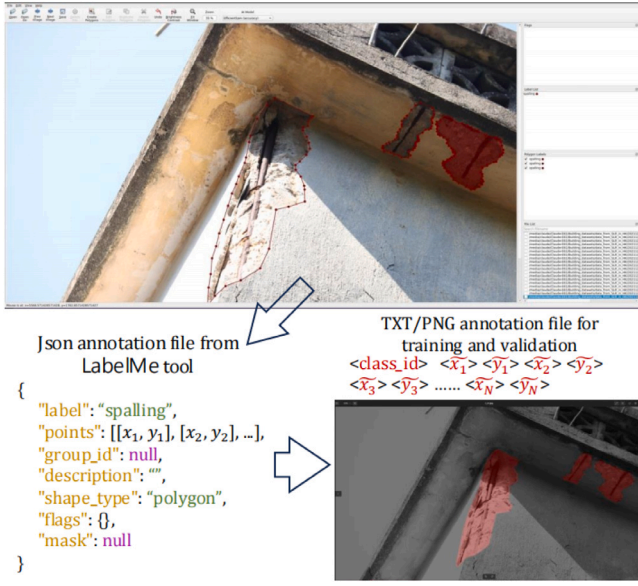


Fig. 5. Interface of LabelMe tools (upper-side) and the format of JSON, TXT and PNG annotation file (lower-side).

together with a professor with relevant domain expertise, and the confirmed decision is then communicated to the annotators as a unified labeling criterion for similar cases. This process is used to improve the practical consistency and reliability of the instance annotations.

Typical borderline cases arise when defect boundaries are affected by non-uniform light-shadow transitions, noisy or stain-like background interference, low-contrast texture patterns, or partial overlap between different defect types. Rather than applying a single fixed size threshold as a universal inclusion rule, we determine inclusion according to whether a defect instance can be clearly recognized and reliably delineated in the native high-resolution image. Defect-like patterns that are extremely faint, visually indeterminate, or highly ambiguous are excluded. In particular, fine-scale cracks are annotated only when their extent can still be visually traced with sufficient confidence. In addition, when crack and spalling regions overlap, we prioritize spalling annotation to avoid inconsistent double labeling of the same damaged area.

Since LabelMe exports annotations in JSON format, which is not always directly compatible with common training and validation pipelines, we further convert the labels into both PNG mask images and YOLO-style polygon TXT files while preserving the instance boundaries. For each image, LabelMe produces a JSON file as shown in the lower-left of Fig. 5, where each polygon vertex is stored as a pixel coordinate $[x_i, y_i]$. We convert these coordinates to normalized YOLO polygon coordinates by $\bar{x}_i = x_i/W$ and $\bar{y}_i = y_i/H$, where W and H denote the image width and height, respectively (see the lower-right of Fig. 5). The bounding box for defect localization is defined as the smallest enclosing rectangle of each polygon. Consequently, each image is associated with a YOLO-format TXT label file and a PNG mask, which are used for subsequent training and evaluation.

3.3. Analysis of CUBIT-InSeg dataset

After describing the data collection strategy and the annotation workflow, we further examine the statistical and geometric characteristics of CUBIT-InSeg. The following analysis focuses on two aspects: the overall dataset statistics together with the spatial distribution of defect instances, and the size distributions of foreground and background boxes, which together help reveal the scale diversity and practical complexity of the dataset.

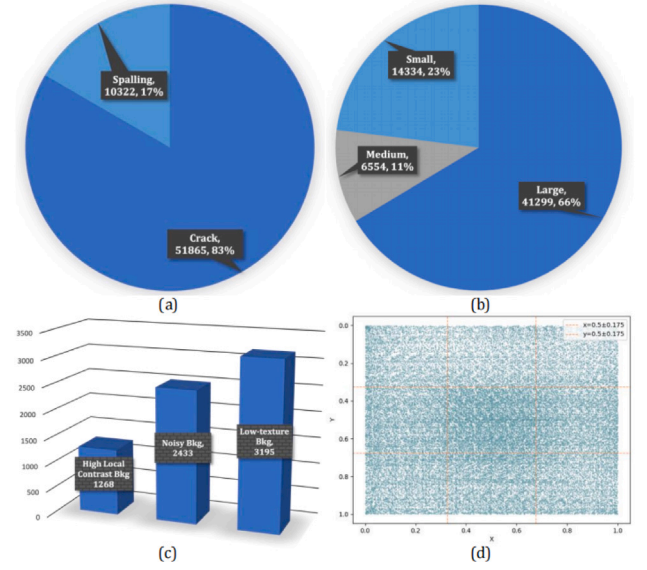


Fig. 6. (a) Defect categories; (b) Defect instance area-ratio; (c) Background scenarios of data collection; (d) Defect instances position distribution.

3.3.1. Statistics and target position distribution analysis of CUBIT-InSeg dataset

Fig. 6 summarizes the dataset statistics and instance distribution of CUBIT-InSeg. In the 6996 images, we annotated 62,187 defect instances, including 51,865 crack instances (83%) and 10,322 spalling instances (17%) (shown in Fig. 6(a)). The large gap between the two categories further reflects the scarcity of spalling in real-world façade inspection. Notably, spalling is typically associated with more severe structural deterioration and potential safety hazards; therefore, despite being less common than cracks, it merits particular attention and motivates dedicated data acquisition efforts. We further categorize instances by their pixel-area ratio within the image (Fig. 6(b)): small (<5%), medium (5%–10%), and large (>10%). Large instances dominate the dataset (41,299; 66%), followed by small instances (14,334; 23%) and medium instances (6,554; 11%).

Fig. 6(c) reports the background scenarios demonstrated in Fig. 3. Low-texture backgrounds constitute the largest portion (3195 images), followed by noisy backgrounds (2433 images), with the remainder belonging to locally high-contrast conditions induced by illumination and shadows. Overall, the diverse and challenging scene settings in CUBIT-InSeg dataset provide a solid basis for cross-domain evaluation and real-world deployment tests.

Beyond instance counts, the spatial distribution of targets is another critical factor in evaluating an instance segmentation dataset. The placement of defects within images influences a model's spatial awareness, which represents its ability to recognize objects across different locations, and directly affects robustness to target offset or positional shifts. In practical UAV inspection scenarios, defects often appear sparsely and unpredictably across the façade. Models trained on datasets with well-distributed targets are therefore more capable of detecting defects located near image borders, in corners, or in unconventional positions. Fig. 6(d) visualizes the scatter plot of ground-truth bounding-box centers for all defect instances. The distribution is reasonably uniform across the image plane, with a moderate concentration near the central region: approximately 63% of all targets (39,078 instances) fall within the normalized range $[x, y] \in \{0.5 \pm 0.175\}$. Such a distribution ensures that models trained on our CUBIT-InSeg dataset develop strong spatial generalization capabilities and exhibit higher robustness when deployed in real-world UAV inspection scenarios.

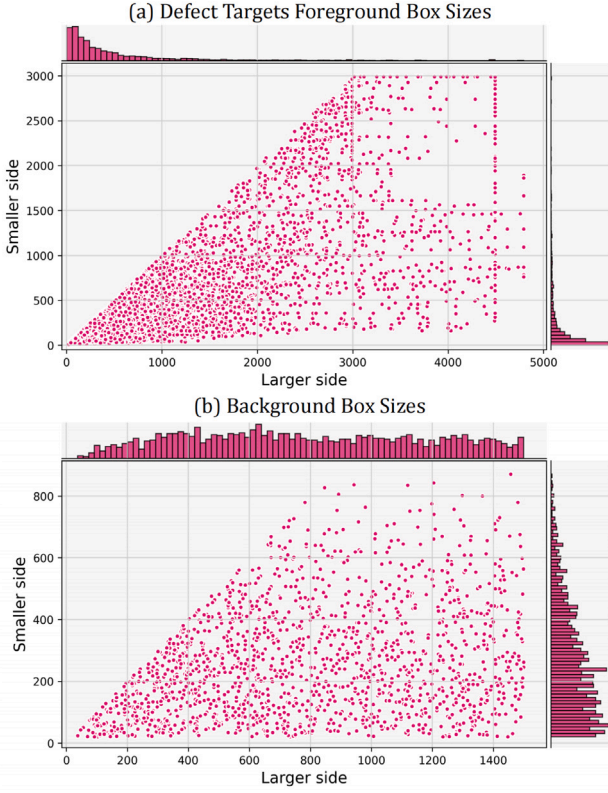


Fig. 7. (a) Distribution of annotated bounding box sizes for defects; (b) Distribution of sizes for sampled non-overlapping background bounding boxes.

3.3.2. Foreground and background box sizes analysis of CUBIT-InSeg dataset

To further characterize the geometric variability of CUBIT-InSeg, Fig. 7(a) illustrates the size distribution of ground-truth defect bounding boxes. The defects span an exceptionally wide scale range: crack instances form thin, elongated structures with short sides sometimes below 5 pixels, whereas spalling regions may exceed 4800 pixels along their longer dimension. The strong concentration of points near the lower boundary of the “Smaller side” axis reflects the high anisotropy and extreme slenderness of cracks, while the marginal histograms reveal a long-tailed distribution covering both microscopic and very large façade defects. This pronounced scale diversity poses a significant challenge for both detection and segmentation.

Fig. 7(b) shows the distribution of background boxes, which are randomly sampled non-defect patches used as a reference for contrastive analysis. These background patches are intentionally constrained to moderate sizes so that they remain representative of local non-defect regions. The empty region in the lower-left corner is expected: background boxes smaller than 20 pixels on either side are excluded, corresponding to the 20-pixel receptive field induced by the 32×down-sampling operation (P5-stage) in common instance segmentation models (input size 640/32 = 20). Patches below this scale do not constitute meaningful semantic context for the model and are therefore omitted. Compared with the highly heterogeneous foreground distribution, the background boxes exhibit a more compact and uniform pattern, highlighting the intrinsic difficulty of detecting tiny cracks that may occupy less than 0.01% of the image while simultaneously handling large-scale structural defects. Together, these characteristics underscore the geometric richness and real-world complexity embodied in the CUBIT-InSeg dataset.

4. Physical quantification and DT integration

While the previous section focuses on dataset establishment and analysis, the value of CUBIT-InSeg also lies in its compatibility with physically grounded façade assessment. In this section, we describe how pixel-level defect segmentations can be converted into physically interpretable measurements and further integrated into a DT-oriented façade representation. Benefiting from the equal-distance imaging setup and calibrated geometric conversion, defect attributes such as crack length, crack width, and spalling area can be estimated in consistent physical units and then registered to corresponding façade elements in the reconstructed model. This enables the transition from image-level defect recognition to component-level condition description and severity-aware defect management.

4.1. Physical-based defect measurement from pixel-level segmentation

Given the equal-distance UAV imaging and automatic flight planning strategy described in the previous subsection, all façade images in CUBIT-InSeg are captured at a prescribed, nearly constant standoff distance d_{target} from the building surface. This imaging geometry enables pixel-level defect instance segmentations to be directly converted into physically meaningful measurements, which are later used for DT-based defect quantification with the workflow in Fig. 8.

Let I_i denote the i th UAV image captured at distance d_{target} with camera focal length f and pixel pitch δ_p (physical size of one pixel on the sensor). Under the pinhole camera model, the ground-sample distance (GSD) on the façade plane can be approximated as

$$\text{GSD} = \frac{d_{\text{target}}}{f} \delta_p, \quad (3)$$

where f is the camera focal length and δ_p is the pixel pitch. As for the equal-distance imaging strategy, the GSD is approximately constant across all images and within each façade patch. In practice, images with excessive lens distortion or misalignment are filtered out. This ensures that the remaining images provide a reliable basis for metric conversion and DT-based defect quantification, while the approximately constant GSD allows pixel measurements to be converted to physical units for cracks and spalling.

To assess the effect of deviations from the equal-distance assumption, we consider a simple sensitivity analysis. If the realized standoff distance deviates from d_{target} by a relative factor η , then crack width and spalling area are perturbed by $\Delta w/w = \eta$ and $\Delta A/A = (1 + \eta)^2 - 1$, respectively. For example, $\pm 5\%$ and $\pm 10\%$ distance deviations lead to approximately $\pm 5\%$ and $\pm 10\%$ errors in crack width, and $\pm 10.25\%$ and $\pm 21\%$ errors in spalling area at the individual-defect level. These cases are intended as conservative sensitivity bounds: for typical inspection distances above 10 m, a $\pm 10\%$ deviation would already correspond to about a 1 m standoff error, which is unusually large and not representative of normal planned operation. Moreover, the final engineering assessment is based on aggregated component-level indicators and interval-based SI grading rather than on a single raw defect measurement, so the practical impact on severity assessment is typically much smaller than the defect-level upper-bound sensitivity alone may suggest.

Using the instance-level segmentation model trained on CUBIT-InSeg, each image I_i is associated with a set of defect instances $\mathcal{M}_i = \{\Omega_i^{(k)}\}_{k=1}^{N_i}$, where $\Omega_i^{(k)} \subset \mathbb{Z}^2$ denotes the pixel set of the k th defect instance (either crack or spalling) in image I_i .

For crack-type instances, we first compute a skeletonized representation to decouple crack *length* and *width*. Let $\Gamma_i^{(k)}$ be the skeleton pixels of instance $\Omega_i^{(k)}$, extracted by a standard thinning algorithm, and let $\mathbf{t}(\mathbf{p})$ be the unit tangent direction along the skeleton at pixel $\mathbf{p} \in \Gamma_i^{(k)}$. The corresponding normal direction is $\mathbf{n}_\perp(\mathbf{p}) = \mathbf{R}_{90^\circ} \mathbf{t}(\mathbf{p})$, where $\mathbf{R}_{90^\circ}$ rotates a 2D vector by 90° .

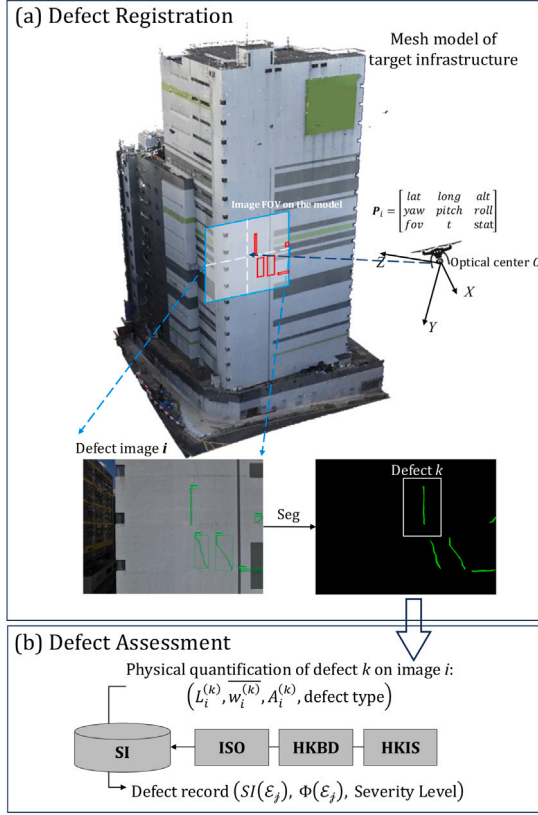


Fig. 8. Physical-based façade defect quantification workflow: (a) Defect registration; (b) Defect assessment.

Crack width For each skeleton pixel $\mathbf{p}$, we count the number of consecutive crack pixels along $\mathbf{n}_\perp(\mathbf{p})$ that remain inside $\Omega_i^{(k)}$: $N_\perp(\mathbf{p}) = |\{\mathbf{q} \in \Omega_i^{(k)} \mid \mathbf{q} \text{ on the normal line } \mathbf{p}\}|$.

The local physical crack width at $\mathbf{p}$ is then

$$w_i^{(k)}(\mathbf{p}) = N_\perp(\mathbf{p}) \text{ GSD}. \quad (4)$$

An instance-level crack width can be defined as the mean or maximum of $w_i^{(k)}(\mathbf{p})$ over all $\mathbf{p} \in \Gamma_i^{(k)}$, e.g.

$$\bar{w}_i^{(k)} = \frac{1}{|\Gamma_i^{(k)}|} \sum_{\mathbf{p} \in \Gamma_i^{(k)}} w_i^{(k)}(\mathbf{p}). \quad (5)$$

Crack length Similarly, the crack length is obtained by summing the physical distances between adjacent skeleton pixels along $\Gamma_i^{(k)}$. Let $\Gamma_i^{(k)} = \{\mathbf{p}_1, \dots, \mathbf{p}_{L_k}\}$ be ordered along the crack centerline; the physical length is approximated as

$$L_i^{(k)} \approx \sum_{j=1}^{L_k-1} \|\mathbf{p}_{j+1} - \mathbf{p}_j\|_2 \text{ GSD}. \quad (6)$$

Spalling area For spalling-type instances, the primary geometric descriptor at the image level is the defect area. Given an instance mask $\Omega_i^{(k)}$ labeled as spalling, its area in pixels is simply $|\Omega_i^{(k)}|$; the corresponding physical area is

$$A_i^{(k)} = |\Omega_i^{(k)}| \text{ GSD}^2. \quad (7)$$

Additional shape descriptors, such as equivalent diameter, aspect ratio, or compactness, can be derived from the pixel mask and converted to physical units by scaling lengths with GSD. These descriptors form a physically consistent feature set for spalling, which is particularly important given its higher severity in façade safety assessment compared with cracks.

The above procedure yields, for each defect instance $\Omega_i^{(k)}$, a collection of physically interpretable measurements, e.g., $(L_i^{(k)}, \bar{w}_i^{(k)}, A_i^{(k)})$, defect type), all expressed in metric units under the equal-distance imaging assumption. These image-level measurements are aggregated at façade- or component-level (e.g., by grouping instances within the same façade panel or elevation zone), providing a compact statistical description of defect conditions. Since the final assessment is based on aggregated defect statistics rather than a single local measurement, moderate random perturbations in individual width, length, or area estimates are partly smoothed out at the component level.

4.2. DT-oriented defect quantification

Building upon the physically measurable crack and spalling descriptors derived from pixel-level segmentations, this section integrates these measurements into a DT representation of the façade. The objective is to transform 2D image-based defect indicators into façade-element-level condition assessments that are physically interpretable and consistent with international building pathology standards, including BS ISO 15686-7:2017 and the inspection guidelines issued by the Hong Kong Buildings Department (BD) and the Hong Kong Institute of Surveyors (HKIS).

Let the façade DT be composed of M surface or BIM elements $\{\mathcal{E}_j\}_{j=1}^M$. Each defect instance detected in images is associated with one or more façade elements based on its image footprint and the camera field-of-view at the time of acquisition.

For each defect instance k detected in image I_i , the set of physically measurable attributes is denoted as

$$\Phi_i^{(k)} = \{L_i^{(k)}, \bar{w}_i^{(k)}, A_i^{(k)}, \text{type}\}, \quad (8)$$

where $L_i^{(k)}$ and $\bar{w}_i^{(k)}$ denote the crack length and mean geometric width, respectively, $A_i^{(k)}$ denotes the spalling area, and $\text{type} \in \{C, S\}$ indicates crack or spalling.

A defect instance is assigned to façade element $\mathcal{E}_j$ if it appears within the region of the façade imaged when capturing I_i . Let $\mathcal{K}_j$ denote the set of all defect instances associated with $\mathcal{E}_j$. The aggregated defect state of element $\mathcal{E}_j$ is therefore

$$\Phi(\mathcal{E}_j) = \bigcup_{k \in \mathcal{K}_j} \Phi_i^{(k)}. \quad (9)$$

For practical façade assessment, the following element-level statistics are computed:

$$L_C(\mathcal{E}_j) = \sum_{k \in \mathcal{K}_j, \text{type}=C} L_i^{(k)}, \quad (10)$$

$$\bar{w}_C(\mathcal{E}_j) = \frac{1}{|\mathcal{K}_j^C|} \sum_{k \in \mathcal{K}_j^C} \bar{w}_i^{(k)}, \quad (11)$$

$$A_S(\mathcal{E}_j) = \sum_{k \in \mathcal{K}_j, \text{type}=S} A_i^{(k)}, \quad (12)$$

where $\mathcal{K}_j^C$ and $\mathcal{K}_j^S$ denote the crack and spalling instances, respectively. These quantities constitute the defect signature of each façade component and serve as the input for severity grading and maintenance prioritization.

According to BS ISO 15686-7:2017 [16], the Hong Kong Buildings Department [17], and the Hong Kong Institute of Surveyors (HKIS) [18], cracks and spalling are both regarded as safety-relevant façade defects. Cracks may lead to moisture ingress, reinforcement corrosion, and progressive deterioration, while spalling is classified as a high-risk defect due to the potential for concrete detachment and falling hazards.

To align with these standards while accounting for the characteristics of UAV-based image measurements, a unified severity index (SI) is defined for each façade element:

$$SI(\mathcal{E}_j) = \alpha_L \tilde{L}_C(\mathcal{E}_j) + \alpha_W \tilde{W}_C^{\text{eff}}(\mathcal{E}_j) + \beta_S \tilde{A}_S(\mathcal{E}_j), \quad (13)$$

Table 3

Severity classification aligned with the proposed SI and ISO/HK inspection practice.

Severity level	SI range	Crack extent (Length)	Effective opening (Qualitative)	Recommended action
Low	$SI < 0.25$	Short, isolated cracks (≤ 0.5 m)	Narrow/visually minor	Routine monitoring
Moderate	$0.25 \leq SI < 0.50$	Multiple or medium-length cracks	Moderate opening	Preventive repair planning
Severe	$0.50 \leq SI < 0.75$	Long or connected crack patterns	Large effective opening	Urgent maintenance required
Critical	$SI \geq 0.75$	Extensive or through-façade cracking	Very large opening/detachment risk	Immediate action and safety measures

where $\alpha_L + \alpha_W + \beta_S = 1$ and $\alpha_L > \alpha_W$, reflecting the engineering observation that crack extent is often more indicative of façade-level risk than localized width measurements.

Here, $\tilde{L}_C$ denotes the normalized total crack length, $\tilde{W}_C^{\text{eff}}$ denotes an effective crack opening indicator derived from geometric width using a soft, logarithmic mapping to avoid over-sensitivity to large apparent widths in image space, and $\tilde{A}_S$ denotes the normalized spalling area. All normalized terms take values in $[0, 1]$.

Based on the computed SI, each façade element is classified into one of four condition levels consistent with ISO 15686 and Hong Kong inspection practice, as summarized in Table 3. The classification emphasizes crack extent and effective opening in combination, while reserving the highest severity levels for extensive or potentially hazardous defect patterns.

The final DT representation stores, for each façade element $\mathcal{E}_j$, the numerical SI, the underlying physical defect metrics, and the corresponding condition level:

$$(\mathcal{SI}(\mathcal{E}_j), \Phi(\mathcal{E}_j), \text{Severity Level}). \quad (14)$$

This enables quantitative visualization of façade conditions, time-series monitoring across repeated UAV inspections, and risk-informed maintenance prioritization within the DT environment.

5. Benchmark evaluation experiments of the proposed CUBIT-InSeg dataset

To comprehensively evaluate the proposed CUBIT-InSeg dataset, we trained an extensive suite of deep learning models, encompassing 17 model families and more than 80 individual networks, including convolution-based architectures (Starnet [54], FasterNet [55], MobileNetV4 [56], EMO [57], ConvNeXtV2 [58]), transformer-based architectures (Swin-Transformer [59], CSwin-Transformer [60], RepViT [61], EfficientViT [62]), and YOLO variants (YOLOv8 [38], YOLOv9 [19], YOLOv10 [63], YOLOv11 [64], YOLOv12 [65], YOLOv13 [66], Mamba-YOLO [67], Hyper-YOLO [39]). These SOTA approaches cover a wide spectrum of design paradigms and represent the leading techniques in modern object detection and instance segmentation. Leveraging such architectural diversity allows us to establish a comprehensive benchmark and to compare how different model families behave under a unified evaluation setting on CUBIT-InSeg. This helps position CUBIT-InSeg as a standardized benchmark resource rather than attributing performance differences to the dataset itself.

For evaluation metrics, we utilize mean Average Precision (mAP) by following the widely adopted MS COCO [10] instance segmentation task, reporting bounding GT-Box mAP_{0.5} and mAP_{0.5:0.95} for objects localization, as well as mask mAP_{0.5} and mAP_{0.5:0.95} for objects segmentation. Adopting these widely accepted metrics also aligns our dataset with common object-centric benchmarks, thereby enhancing its comparability and general applicability within the broader community.

5.1. Experimental setup

All experiments, including both the benchmark evaluation on CUBIT-InSeg and the cross-domain validation on external datasets, are conducted on an Ubuntu 22.04 workstation equipped with an Intel i9-13900K CPU and dual NVIDIA RTX 4090 GPUs. All models are trained for 300 epochs with a batch size of 16. To ensure fair comparison,

Table 4

Key training and validation/test settings of experiments.

Input images size	640/512/416 CUBIT-InSeg/Highway-Crack/Crack-Seg
Epochs	300
Batch size	16
Optimizer type	SGD
Learning rate schedule	Linear
Initial learning rate α	1e-2
Final LR factor (lrf)	1e-4
Momentum β	0.937
Weight decay ϕ	5e-4
Warmup epochs	3.0
Warmup momentum	0.8
Warmup bias learning rate	0.1
Loss coefficients $\lambda_{cls}, \lambda_{box}, \lambda_{dfl}$	0.5, 7.5, 1.5
HSV-H augmentation	0.015
HSV-S augmentation	0.7
HSV-V augmentation	0.4
Translation	0.1
Scaling	0.5
Horizontal flip	0.5
Vertical flip	0.5
Mosaic	1.0
MixUp	0.5
Copy-paste	0.0
Degrees	0.0
Shear	0.0
Perspective	0.0
Random seed	0
Deterministic training	Enabled

no model is initialized with pre-trained weights from generic object-detection or defect-related datasets (e.g. MS COCO [10]). Stochastic gradient descent (SGD) is used as the optimizer, and the key training, augmentation, and reproducibility settings are summarized in Table 4. In addition, all experiments are conducted with a fixed random seed (seed=0), and deterministic computation is enabled where supported by the Ultralytics framework.

During inference, non-maximum suppression (NMS) [68] is applied to remove redundant detections. For instance segmentation, although the segmentation mask is ultimately evaluated using mask-level IoU, the suppression process still relies on the GT-Box IoU, consistent with the procedure defined in Algorithm 1. Given an initial list of detected bounding boxes B and their confidence scores S , NMS iteratively selects the box with the highest confidence score, denoted as M , and adds it to the final detection set D . All remaining boxes $b_i \in B$ whose box IoU with M exceeds the suppression threshold N_t are removed along with their corresponding scores. Although mask-level IoU is later used to compute mask-based evaluation metrics such as Mask_AP_{0.5} and Mask_AP_{0.5:0.95}, the suppression step remains box-based to ensure consistency and computational efficiency. These mask-based metrics evaluate the quality of the predicted segmentation masks by measuring their pixel-level agreement with the ground-truth shapes, thus reflecting the accuracy of instance-level defect delineation.

The IoU criterion evaluates the overlap ratio between a predicted box/mask (P-Box/P-Mask) and the ground-truth box/mask (GT-Box/GT-Mask). We adopt a confidence threshold of 0.25 and an IoU threshold of 0.7, which follows the setting of Ultralytics framework unless other noted. A visual illustration of the IoU of computation is provided in Fig. 9.

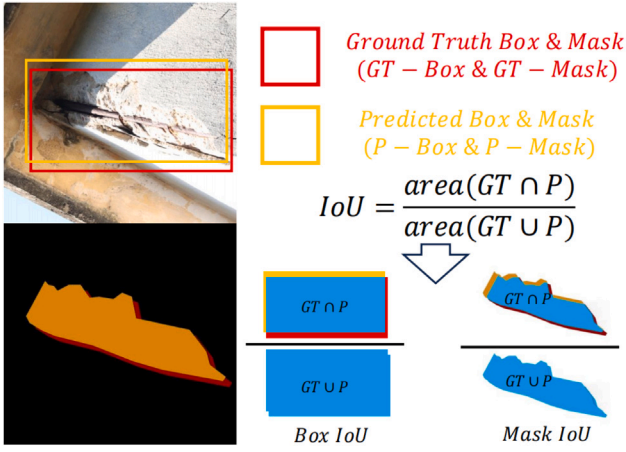


Fig. 9. Visualization of Intersection-over-Union (IoU). Red and orange represent the ground-truth bounding box/mask (GT-Box/GT-Mask) and predicted box/mask (P-Box/P-Mask) of this spalling sample, respectively. In IoU equation, the denominator symbolizes the union of the GT and the P, which is represented by a blue area. The overlapping area of the GT and the P, which denoted their intersection, is also indicated by the blue part.

For benchmark training and testing on CUBIT-InSeg, the dataset is split into training (5596 images, 80%), validation (699 images, 10%), and test (701 images, 10%) sets, with all images resized to 640×640 . Among the aforementioned SOTA models, both convolution-based and transformer-based architectures serve as powerful feature extraction backbones. To ensure a fair and controlled comparison, we integrate all these backbones into the Ultralytics [38,64] framework, which is also used by the YOLO series and its variants. In our unified benchmark, only the backbone is replaced across different model variants, while the neck and detection/segmentation heads are kept consistent under the same Ultralytics (YOLOv11-style) design. Specifically, the multi-scale feature maps extracted by each backbone are aligned and connected through the corresponding P3, P4, and P5 levels to the same shared neck and head for subsequent feature fusion, detection, and instance segmentation. As a result, the primary variation among compared models comes from the backbone itself, whereas the neck, head, prediction pipeline, training recipe, and inference protocol remain unified. Such a design is intended not to fully preserve every original architecture-specific component, but to establish a controlled, reproducible, and horizontally comparable benchmark setting for CUBIT-InSeg. For a dataset work, this unified protocol is important because it reduces confounding factors introduced by heterogeneous implementations and allows performance differences to more clearly reflect how different backbone families respond to the dataset. Unless otherwise noted, data preprocessing settings (including image resizing, normalization, and anchor settings) follow the default Ultralytics pipeline implementation. Accordingly, the benchmark results should be interpreted primarily as differences among model architectures under a common protocol, while the contribution of CUBIT-InSeg lies in providing the high-resolution, instance-level, and reproducible evaluation setting (see Fig. 10).

5.2. Evaluation metrics

We evaluate all models using the standard COCO evaluation protocol, which is applicable to both object detection and segmentation. For each predicted instance, the IoU between the prediction and its corresponding ground truth is computed—using bounding boxes for detection and pixel-level masks for segmentation. Based on IoU, a prediction is classified as a *True Positive* (TP), *False Positive* (FP), or *False Negative* (FN).

Algorithm 1: Non-maximum suppression (NMS) procedure used in instance segmentation pipeline

Input: The **input** initial detection boxes B , the corresponding confidence scores S , and the IoU suppression threshold N_t .
Output: The **output** final selected boxes D and their corresponding scores S .

```

1  $D \leftarrow \emptyset$ 
2 while  $B \neq \text{empty}$  do
3   Select the maximum value in the set of  $S$ , and give this value to
    $m, m \leftarrow \text{argmax}(S)$ 
4    $M \leftarrow b_m$ 
5    $D \leftarrow D \cup M$ 
6    $B \leftarrow B - M$ 
7   for  $b_i$  in  $B$  do
8     if  $\text{iou}(M, b_i) > N_t$  then
9        $B \leftarrow B - b_i; S \leftarrow S - s_i;$ 
10  return  $D, S$ 

```

The fundamental metrics Precision (P) and Recall (R) are defined as:

$$\text{Precision} = \frac{TP}{TP + FP}, \quad \text{Recall} = \frac{TP}{TP + FN}, \quad (15)$$

measuring the correctness of predictions and the ability to recover ground-truth defects, respectively.

To summarize performance across different recall levels, we adopt the COCO-style *Average Precision* (AP), computed by integrating the *Precision-Recall* curve. Following the COCO protocol, AP is evaluated at IoU threshold 0.5 ($AP_{0.5}$) and averaged across ten thresholds from 0.50 to 0.95 ($AP_{0.5:0.95}$). For multi-class tasks, the mAP is obtained by averaging AP over all defect categories.

$$AP = \int_0^1 p(r) dr, \quad mAP = \frac{1}{C} \sum_{c=1}^C AP_c. \quad (16)$$

where c and C represent each defect class and total defect class numbers, respectively.

We report both *Box_mAP* (based on box IoU) and *Mask_mAP* (based on mask IoU). The box-based metrics evaluate localization accuracy, while the mask-based metrics assess pixel-level agreement and boundary fidelity—crucial for infrastructure defect segmentation.

5.3. Benchmarking experiment and analysis

Based on the experimental setup and evaluation metrics described above, we report the benchmark results of more than 80 models on CUBIT-InSeg, together with the corresponding cross-domain zero-shot evaluation results. The following analysis covers three aspects: overall and category-wise instance segmentation performance, the relationship between network attributes and practical efficiency, and zero-shot transferability to external defect datasets.

5.3.1. Overall and category-wise performance analysis of instance segmentation

Table 5 and 6 not only show the overall experimental results but also report the per-defect-type evaluation results tested on our proposed CUBIT-InSeg dataset.

Since all SOTA methods in our benchmark are re-implemented under a unified Ultralytics protocol to ensure fair and reproducible comparison, we first conduct a supplementary ablation on three representative backbones, which are *FasterNet-s* [55], *ConvNeXtV2-tiny* [58], and *Swin-Transformer-Tiny* [59], covering lightweight CNN, modern CNN, and modern Transformer families, by comparing their official MMDetection-based [69] *Mask R-CNN* [35] implementations with our unified re-implementations. As shown in

Table 5
Benchmark results of selected end-to-end models on the test set of CUBIT-InSeg dataset.

Models	#Params(M)↓	FLOPs(G)↓	FPS↑	Box_mAP ^{test} _{0.5} ↑	Box_mAP ^{test} _{0.5:0.95} ↑	Crack (Box)		Spalling (Box)		Mask_mAP ^{test} _{0.5} ↑	Mask_mAP ^{test} _{0.5:0.95} ↑	Crack (Mask)		Spalling (Mask)	
						AP ^{test} _{0.5} ↑	AP ^{test} _{0.5:0.95} ↑	AP ^{test} _{0.5} ↑	AP ^{test} _{0.5:0.95} ↑			AP ^{test} _{0.5} ↑	AP ^{test} _{0.5:0.95} ↑	AP ^{test} _{0.5} ↑	AP ^{test} _{0.5:0.95} ↑
Conv-based Models															
EMO-1M (n) [57]	3.27	13.1	498.35	84.5%	74.4%	71.1%	54.5%	97.9%	94.4%	76.1%	52.7%	54.7%	19.0%	97.4%	86.3%
EMO-2M (s) [57]	9.06	37.1	356.96	88.0%	78.7%	77.6%	61.9%	98.4%	95.5%	78.6%	55.3%	59.6%	22.4%	97.7%	88.2%
EMO-5M (m) [57]	20.46	86.1	242.75	90.4% ₂	82.2%	82.0% ₂	68.0%	98.8% ₂	96.4%	81.6% ₁	57.5%	64.9% ₂	25.2% ₁	98.3% ₁	89.8%
EMO-6M (l) [57]	32.12	149.5	203.45	90.5% ₁	82.4% ₂	82.1% ₁	68.2% ₂	98.8% ₂	96.7% ₁	81.2%	57.9% ₁	64.0%	25.0%	98.3% ₁	90.7% ₁
MobileNetV4-S (s) [56]	10.19	41.8	592.27	78.3%	66.4%	58.9%	40.8%	97.7%	92.0%	71.2%	48.5%	45.5%	13.3%	96.9%	83.7%
MobileNetV4-M (m) [56]	23.27	140.0	340.33	79.1%	67.6%	60.4%	42.3%	97.8%	93.0%	71.3%	48.6%	45.7%	13.7%	97.0%	83.4%
MobileNetV4-L (l) [56]	34.49	154.7	285.14	80.9%	70.0%	64.1%	46.1%	97.6%	93.8%	72.9%	50.5%	48.6%	14.8%	97.1%	86.1%
FasterNet-t0 (n) [55]	4.15	13.1	729.19 ₂	79.7%	68.2%	61.7%	43.2%	97.8%	93.1%	72.2%	49.0%	47.1%	14.1%	97.4%	84.0%
FasterNet-t1 (n) [55]	7.79	21.7	640.52	82.1%	70.6%	66.0%	47.4%	98.2%	93.9%	74.4%	50.3%	51.0%	16.0%	97.8%	84.7%
FasterNet-t2 (n) [55]	15.17	39.4	508.73	82.7%	71.3%	67.4%	48.9%	98.1%	93.7%	74.2%	50.6%	50.7%	15.5%	97.7%	85.8%
FasterNet-s (s) [55]	35.85	100.2	302.47	85.8%	78.0%	72.8%	54.2%	98.8% ₂	95.3%	76.0%	52.3%	53.8%	16.9%	98.1%	87.7%
FasterNet-m (m) [55]	65.57	228.4	160.52	87.6%	77.3%	76.5%	58.8%	98.8% ₂	95.8%	77.9%	53.9%	57.5%	19.2%	98.2% ₂	88.5%
FasterNet-l (l) [55]	109.29	349.8	124.97	88.4%	78.2%	78.0%	60.2%	98.9% ₁	96.2%	78.5%	54.5%	58.8%	19.9%	98.3% ₁	88.9%
Starnet-s0 (n) [54]	2.19	8.9	828.82 ₁	72.9%	61.0%	48.8%	31.9%	97.1%	90.0%	67.6%	46.0%	39.1%	11.2%	96.1%	80.8%
Starnet-s100 (n) [54]	2.69 ₂	10.4 ₂	689.58	76.8%	65.0%	56.0%	38.2%	97.5%	91.8%	69.9%	47.5%	43.1%	12.8%	96.7%	82.2%
Starnet-s150 (n) [54]	3.19	11.2	665.89	77.1%	65.1%	56.7%	38.5%	97.6%	91.7%	70.8%	47.8%	44.7%	13.2%	96.8%	82.4%
Starnet-s1 (s) [54]	8.45	30.8	480.88	82.3%	71.3%	66.5%	48.3%	98.1%	94.2%	74.8%	51.5%	52.1%	17.2%	97.5%	85.9%
Starnet-s2 (m) [54]	16.4	91.6	341.02	85.6%	75.4%	72.5%	55.1%	98.7%	95.6%	77.0%	53.4%	56.3%	18.4%	97.8%	88.3%
Starnet-s3 (l) [54]	21.7	104.3	287.39	86.0%	75.4%	73.3%	55.5%	98.8% ₂	95.3%	77.4%	53.4%	56.8%	18.7%	98.0%	88.1%
Starnet-s4 (x) [54]	43.3	223.0	173.83	86.6%	76.1%	74.4%	56.7%	98.8% ₂	95.5%	78.3%	54.1%	58.6%	19.8%	98.0%	88.4%
ConvNeXtV2-nano (n) [58]	17.91	47.8	210.86	87.6%	79.6%	77.4%	63.8%	97.8%	95.4%	79.1%	55.8%	60.7%	23.0%	97.4%	88.6%
ConvNeXtV2-tiny (s) [58]	35.36	97.7	136.16	87.8%	79.7%	77.5%	63.2%	98.2%	96.2%	79.3%	56.5%	60.7%	22.7%	97.8%	90.4% ₂
ConvNeXtV2-base (m) [58]	103.21	335.6	60.87	89.0%	81.0%	79.7%	65.9%	98.3%	96.2%	80.6%	57.0%	63.2%	24.3%	97.9%	89.7%
ConvNeXtV2-large (l) [58]	217.03	656.9	37.58	90.1%	82.5% ₁	81.8%	68.4% ₁	98.5%	96.7% ₁	81.5% ₂	57.7% ₂	65.1% ₁	25.1% ₂	98.0%	90.4% ₂
Transformer-based Models															
Swin-Transformer-Tiny (n) [59]	29.97	81.5	200.99	82.6%	72.3%	67.4%	50.4%	97.9%	94.2%	74.0%	51.9%	50.7%	17.2%	97.4%	86.5%
Swin-Transformer-Small (s) [59]	55.57	168.8	129.89	86.2%	76.2%	74.1%	57.0%	98.3%	95.3%	76.3%	53.7%	54.7%	18.9%	97.9%	88.5%
Swin-Transformer-Base (m) [59]	62.77	228.0	111.19	87.1%	77.6%	75.5%	59.0%	98.6%	96.1%	77.3%	54.6%	56.4%	19.8%	98.1%	89.5%
Swin-Transformer-Large (l) [59]	66.04	237.4	101.03	87.9%	78.3%	77.2%	60.9%	98.6%	95.8%	78.2%	55.0%	58.3%	20.4%	98.1%	89.6%
CSwin-Transformer-Tiny (n) [60]	23.94	74.4	210.68	81.8%	69.2%	66.3%	46.2%	97.2%	92.2%	72.9%	49.7%	49.1%	14.4%	96.8%	84.9%
CSwin-Transformer-Small (s) [60]	40.39	129.1	136.11	84.4%	72.8%	71.5%	52.3%	97.4%	93.3%	76.2%	52.1%	55.5%	17.6%	96.8%	86.7%
CSwin-Transformer-Base (m) [60]	90.53	318.7	72.24	86.1%	75.0%	74.7%	56.0%	97.5%	94.1%	77.3%	53.8%	57.9%	18.9%	96.8%	88.7%
CSwin-Transformer-Large (l) [60]	190.15	621.2	42.75	86.4%	75.3%	75.2%	56.7%	97.6%	93.9%	78.5%	54.5%	59.6%	20.1%	97.4%	88.9%
EfficientViT-M0 (n) [62]	3.99	11.8	641.55	79.1%	67.4%	60.3%	41.9%	97.9%	92.9%	71.6%	48.6%	45.7%	13.1%	97.5%	84.2%
EfficientViT-M1 (n) [62]	4.63	16.5	546.71	81.3%	69.3%	64.4%	45.4%	98.1%	93.3%	73.6%	50.1%	50.1%	15.1%	97.2%	85.1%
EfficientViT-M2 (s) [62]	9.81	35.2	474.59	84.2%	73.0%	70.0%	51.3%	98.3%	94.8%	76.0%	52.3%	54.2%	17.4%	97.8%	87.1%
EfficientViT-M3 (m) [62]	19.71	97.8	312.00	87.0%	76.7%	75.2%	57.6%	98.8% ₂	95.9%	78.2%	54.4%	58.3%	19.8%	98.0%	89.0%
EfficientViT-M4 (l) [62]	24.88	109.2	264.32	87.3%	77.2%	75.7%	58.3%	98.8% ₂	96.1%	78.7%	54.7%	59.2%	20.3%	98.2% ₂	89.0%
EfficientViT-M5 (x) [62]	48.60	236.1	172.96	89.0%	79.4%	79.1%	62.2%	98.9% ₁	96.5% ₂	80.0%	56.0%	61.8%	21.8%	98.2% ₂	90.2%
RepViT-m09 (n) [61]	6.68	20.9	504.95	82.5%	71.1%	66.9%	48.8%	98.0%	93.8%	75.2%	51.4%	53.0%	16.9%	97.4%	86.0%
RepViT-m10 (s) [61]	12.5	42.4	384.80	86.2%	75.9%	73.7%	56.3%	98.8% ₂	95.4%	78.1%	54.3%	58.1%	20.4%	98.1%	88.3%
RepViT-m11 (m) [61]	21.1	105.2	294.08	87.0%	77.2%	75.2%	58.0%	98.9% ₁	96.4%	78.7%	54.8%	59.2%	20.6%	98.1%	89.0%
RepViT-m15 (l) [61]	30.2	129.7	239.38	87.3%	77.2%	75.9%	58.5%	98.6%	95.8%	78.8%	54.8%	59.5%	20.3%	98.1%	89.4%
RepViT-m23 (x) [61]	59.3	281.0	129.65	87.7%	77.5%	76.4%	59.1%	98.9% ₁	95.9%	79.2%	54.9%	60.2%	20.8%	98.1%	89.0%

(1) ↑ (↓) indicates that larger (smaller) values lead to better (worse) results. The best are in orange, the second best are in blue.

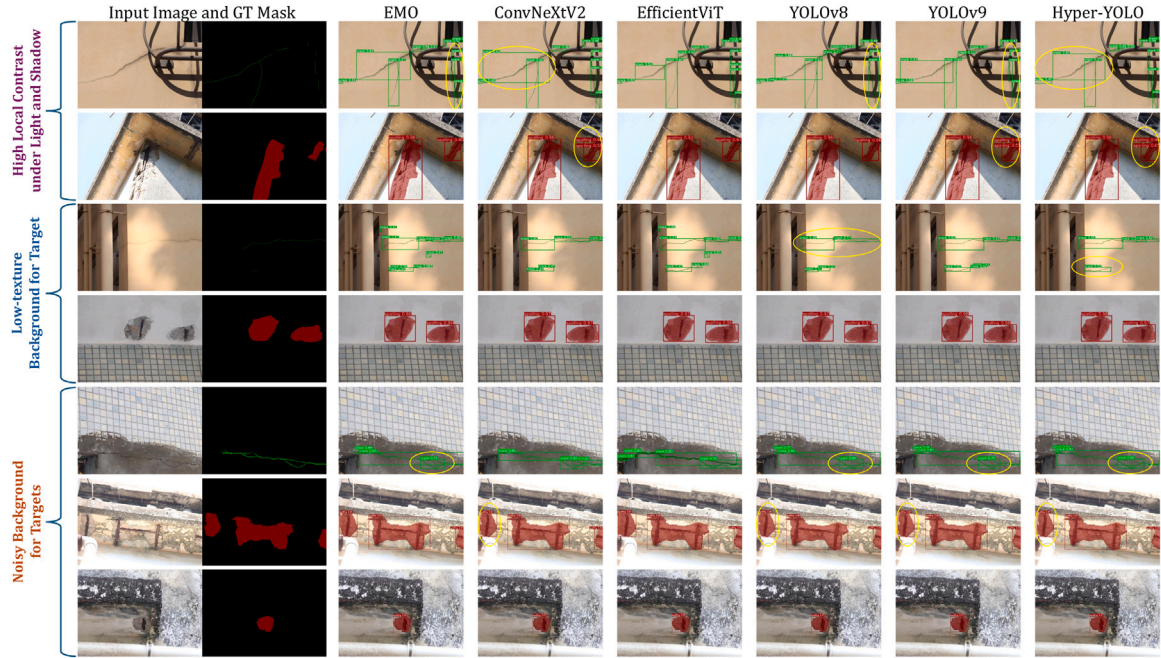


Fig. 10. Qualitative results of the test-set from the proposed CUBIT-InSeg dataset. Yellow ovals highlight the regions where the models instance segmentation effects are relatively good.

Table 6

Benchmark results of selected single-stage real-time models on the test set of CUBIT-InSeg dataset.

Models	#Params(M)↓	FLOPs(G)↓	FPS↑	Box_mAP ^{test} _{0.5} ↑	Box_mAP ^{test} _{0.5:0.95} ↑	Crack (Box)		Spalling (Box)		Mask_mAP ^{test} _{0.5} ↑	Mask_mAP ^{test} _{0.5:0.95} ↑	Crack (Mask)		Spalling (Mask)	
						AP ^{test} _{0.5} ↑	AP ^{test} _{0.5:0.95} ↑	AP ^{test} _{0.5} ↑	AP ^{test} _{0.5:0.95} ↑			AP ^{test} _{0.5} ↑	AP ^{test} _{0.5:0.95} ↑	AP ^{test} _{0.5} ↑	AP ^{test} _{0.5:0.95} ↑
YOLO Series Models															
YOLOv8(8.1)-n [38]	3.25	12.0	835.53 ₂	82.9%	73.0%	68.3%	52.2%	97.5%	93.7%	75.3%	52.0%	53.7%	19.2%	96.9%	84.9%
YOLOv8(8.1)-s [38]	11.78	42.4	545.83	88.4%	79.2%	78.2%	62.6%	98.6%	95.8%	79.6%	55.7%	60.9%	22.7%	98.2%	88.8%
YOLOv8(8.1)-m [38]	27.22	111.0	311.14	90.5%	82.4%	82.3%	68.2%	98.6%	96.5%	81.6%	57.8%	65.2%	25.5%	98.0%	90.0%
YOLOv8(8.1)-l [38]	45.91	220.1	220.34	91.5%	83.8%	84.0%	70.7%	99.0%	96.9%	82.6%	58.7%	66.7%	26.6%	98.5%	90.7%
YOLOv8(8.1)-x [38]	71.72	343.7	145.91	92.0%	85.1%	85.1%	73.1%	98.8%	97.1%	83.0%	59.4%	67.9%	28.1%	98.2%	90.7%
YOLOv9-n [19]	3.09	54.1	536.10	80.7%	70.6%	63.7%	47.6%	97.8%	93.6%	74.0%	52.0%	50.8%	17.5%	97.3%	86.4%
YOLOv9-s [19]	8.52	75.4	408.28	87.6%	78.5%	76.9%	61.5%	98.3%	95.5%	79.6%	55.9%	61.3%	23.0%	97.9%	88.7%
YOLOv9-m [19]	22.25	131.2	275.07	91.0%	82.9%	83.2%	69.3%	98.8%	96.5%	81.4%	57.8%	64.9%	25.4%	98.0%	90.2%
YOLOv9-c [19]	27.62	157.6	235.83	91.9%	84.3%	85.0%	71.7%	98.8%	96.8%	82.5%	58.7%	66.8%	26.6%	98.2%	90.9%
YOLOv9-e [19]	59.68	244.4	128.57	92.5%	85.7%	86.0%	74.2%	98.9%	97.3%	83.9%	60.4%	69.6%	29.5%	98.3%	91.2%
YOLOv10-n [63]	2.84	11.7	836.54 ₁	84.4%	74.8%	71.2%	55.7%	97.7%	93.9%	77.5%	53.6%	57.5%	21.1%	94.4%	86.1%
YOLOv10-s [63]	9.17	40.5	530.40	89.3%	80.6%	80.1%	65.3%	98.5%	96.0%	80.7%	56.7%	63.3%	24.5%	98.0%	88.9%
YOLOv10-m [63]	19.33	101.4	305.73	90.7%	82.8%	82.8%	69.2%	98.7%	96.3%	82.6%	58.3%	67.0%	26.8%	98.2%	89.8%
YOLOv10-b [63]	25.48	166.6	254.87	90.9%	83.0%	83.2%	69.4%	98.6%	96.6%	82.2%	58.4%	66.4%	26.7%	98.1%	90.1%
YOLOv10-l [63]	30.79	194.9	224.79	90.8%	83.2%	82.7%	69.5%	98.9%	96.9%	82.3%	58.7%	66.2%	26.9%	98.4%	90.5%
YOLOv10-x [63]	39.52	276.9	154.25	91.4%	84.2%	83.9%	71.3%	99.0%	97.2%	83.0%	59.3%	67.5%	27.9%	98.4%	90.7%
YOLOv11(8.3)-n [64]	2.83	10.2 ₂	723.71	78.9%	66.8%	60.1%	41.2%	97.8%	92.4%	71.4%	48.3%	45.8%	13.5%	97.0%	83.1%
YOLOv11(8.3)-s [64]	10.07	35.3	466.25	85.8%	74.8%	73.2%	54.8%	98.3%	94.8%	77.8%	53.3%	57.7%	19.3%	98.0%	87.2%
YOLOv11(8.3)-m [64]	22.34	123.0	283.47	88.8%	78.5%	78.7%	61.3%	98.9%	95.7%	79.2%	55.0%	60.2%	21.2%	98.3%	88.8%
YOLOv11(8.3)-l [64]	27.59	141.9	240.49	89.1%	79.7%	79.3%	62.9%	98.9%	96.5%	80.0%	56.1%	61.7%	22.4%	98.3%	89.8%
YOLOv11(8.3)-x [64]	62.01	318.5	131.13	90.9%	81.8%	82.8%	66.8%	99.1%	96.7%	81.1%	57.2%	63.7%	23.9%	98.4%	90.4%
YOLOv12-n [65]	2.81 ₂	10.2 ₂	727.60	79.9%	68.2%	62.6%	44.0%	97.3%	92.5%	72.4%	49.2%	47.9%	14.6%	96.9%	83.9%
YOLOv12-s [65]	9.89	35.2	450.47	85.1%	74.1%	72.0%	53.6%	98.2%	94.7%	77.0%	52.5%	56.0%	18.4%	97.9%	86.6%
YOLOv12-m [65]	22.41	122.4	248.22	87.0%	76.2%	75.4%	57.0%	98.5%	95.3%	77.7%	53.7%	57.3%	19.1%	98.1%	88.4%
YOLOv12-l [65]	28.65	143.9	191.08	88.6%	78.5%	78.4%	60.8%	98.9%	96.3%	79.2%	55.0%	60.2%	20.8%	98.2%	89.2%
YOLOv12-x [65]	64.22	322.6	109.11	88.7%	78.9%	78.5%	61.5%	98.9%	96.4%	78.8%	55.2%	59.3%	20.6%	98.3%	89.7%
YOLOv13-n [66]	2.70 ₁	10.0 ₁	614.38	83.8%	75.4%	70.2%	56.7%	97.4%	94.1%	77.1%	53.9%	57.1%	21.2%	97.1%	86.5%
YOLOv13-s [66]	9.66	34.0	367.35	89.0%	81.6%	79.8%	66.9%	98.2%	96.3%	80.5%	57.0%	63.4%	25.3%	97.7%	88.8%
YOLOv13-l [66]	29.19	139.6	140.64	91.0%	84.0%	83.4%	71.2%	98.7%	96.8%	82.4%	58.9%	66.8%	27.2%	98.1%	90.5%
YOLOv13-x [66]	67.64	311.5	84.86	91.8%	84.3%	84.9%	71.9%	98.8%	96.7%	83.0%	59.1%	67.8%	27.5%	98.2%	90.8%
Mamba-YOLO-T [67]	5.92	16.2	243.10	83.3%	72.3%	68.4%	50.2%	98.2%	94.5%	75.6%	52.2%	53.2%	17.5%	98.0%	87.0%
Mamba-YOLO-B [67]	21.15	58.5	121.19	87.5%	76.7%	76.0%	57.5%	98.9%	95.9%	77.2%	54.0%	55.8%	18.3%	98.5%	89.8%
Mamba-YOLO-L [67]	56.33	175.9	45.08	88.5%	78.6%	78.0%	60.9%	98.9%	96.2%	80.0%	55.8%	61.6%	21.9%	98.4%	88.8%
Hyper-YOLO-n [39]	3.87	13.4	742.44	85.4%	76.5%	72.8%	58.1%	98.1%	94.8%	77.6%	54.0%	57.4%	21.2%	97.7%	86.8%
Hyper-YOLO-s [39]	14.17	47.8	442.49	89.7%	81.4%	80.7%	66.4%	98.8%	96.5%	80.9%	56.7%	63.7%	24.2%	98.1%	89.2%
Hyper-YOLO-m [39]	32.08	123.1	249.75	91.5%	84.3%	84.3%	71.7%	98.8%	96.9%	82.9%	58.7%	67.5%	27.0%	98.2%	90.3%
Hyper-YOLO-l [39]	54.40	246.3	170.52	92.0%	85.4%	85.3%	73.6%	98.8%	97.1%	83.6%	59.7%	69.0%	28.2%	98.3%	91.2%
Hyper-YOLO-x [39]	94.97	384.4	114.87	92.4%	85.7%	85.8%	74.3%	99.0%	97.1%	83.9%	59.9%	69.3%	28.6%	98.4%	91.2%

(1) ↑ (↓) indicates that larger (smaller) values lead to better (worse) results. The best in orange, the second best are blue.

Table 7

Comparison experiment on official and unified implementations using three representative backbones.

Models	Frameworks	#Params (M)↓	FLOPs (G)↓	Box (Detection)		Mask (Segmentation)	
				mAP ^{test} _{0.5} ↑	mAP ^{test} _{0.5:0.95} ↑	mAP ^{test} _{0.5} ↑	mAP ^{test} _{0.5:0.95} ↑
FasterNet-s [55]	Official	48.60	144.0	67.7%	53.1%	47.9%	37.8%
	Unified (Ours)	35.85	100.2	79.9%	61.7%	76.0%	54.2%
ConvNeXtV2-tiny [58]	Official	47.67	143.0	72.1%	59.8%	47.4%	39.1%
	Unified (Ours)	35.36	97.7	87.8%	79.7%	79.3%	56.5%
Swin-Transformer-tiny [59]	Official	47.38	145.0	76.5%	65.2%	49.2%	40.4%
	Unified (Ours)	29.97	81.5	82.6%	72.3%	74.0%	51.9%

(1) “Official” refers to the MMDetection-based Mask R-CNN implementation with the original backbone. “Unified (Ours)” refers to our re-implementation under the shared Ultralytics protocol.

Table 7, the unified setting does not weaken the effectiveness of these backbones on CUBIT-InSeg; instead, it consistently achieves better accuracy with lower complexity. For instance, **FasterNet-s** improves from 53.1% / 37.8% to 61.7% / 54.2% in detection/segmentation mAP^{test}_{0.5:0.95}, while reducing parameters and FLOPs from 48.60M/144.0G to 35.85M / 100.2G. Similar trends are observed for **ConvNeXtV2-tiny** and **Swin-Transformer-Tiny**. This supplementary ablation study not only supports the fairness of the unified benchmark protocol, but also further demonstrates that CUBIT-InSeg can serve as a stable and discriminative benchmark across different model families.

As the sample data shown in Fig. 10, the detection and segmentation of the cracks are substantially more challenging than spallings: cracks typically appear thin, elongated, and fragmented, with ambiguous boundaries, whereas spalling regions are larger, more coherent, and visually well-defined. And the significant disparity in the number of targets is another factor (shown in Fig. 6). As illustrated in Fig. 11, the Box and Mask AP of Spalling lie much closer to the outer boundary of the radar plots compared with those for Crack, which means that the performance gap among models on spallings is relatively small,

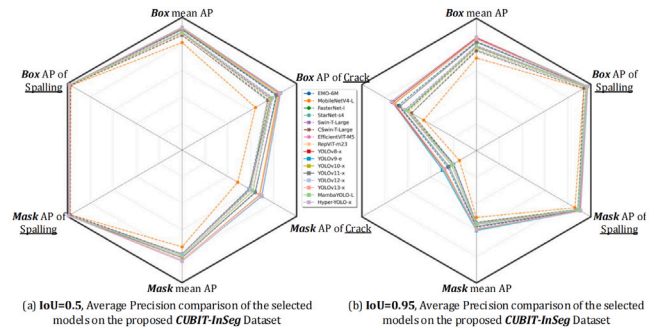


Fig. 11. Comparison of six evaluation metrics at IoU thresholds of 0.5 (a) and 0.5:0.95 (b) using radar plots. Dashed curves correspond to end-to-end models, whereas solid curves correspond to single-stage real-time models.

while cracks exhibit noticeably lower AP values and contribute more significantly to the mAP degradation.

End-to-End Models As shown in Table 5, the conv-based architectures consistently outperform the transformer-based ones while also maintaining noticeably smaller model sizes. This performance advantage is largely driven by their superior ability to handle the more challenging crack category across all four mAP metrics. Benefiting from its conv-based *expanded window multi-head self-attention* (EW-MHSA) module, **EMO-6M** ranks first on Box_mAP_{0.5} and Mask_mAP_{0.5:0.95}, while **EMO-5M** achieves the top score on Mask_mAP_{0.5}. In addition, **ConvNeXtV2-Large**, equipped with the *global response normalization* (GRN) layer, attains the best performance on Box_mAP_{0.5:0.95}.

For spalling category, the performance differences among large models are marginal. Notably, the transformer-based **EfficientViT-M5** performs competitively with conv-based models such as **EMO-6M** and **FasterNet-1** on both Box and Mask_AP_{0.5}.

Single-Stage Real-Time Models As presented in Table 6, the YOLO family and its representative variants deliver strong performance despite their relatively compact model sizes. **YOLOv9-e**, supported by a *plug-and-play auxiliary training branch*, ranks first in three out of four crack-related AP metrics and ultimately achieves the best overall results across four box and mask mAP scores. The second-best performer is **Hyper-YOLO-x**, which benefits from the *hypergraph computation empowered semantic collecting and scattering* (HGCS) framework. **Hyper-YOLO-x** achieves the highest Box_mAP_{0.5:0.95} on crack, but its Mask_mAP_{0.5:0.95} is 0.9% lower than that of **YOLOv9-e**, which is a notable margin at such a stringent IoU threshold.

For the *spalling* category, performance differences are again small. At IoU = 0.5, **YOLOv11-x** gets the highest Box_AP, while **Mamba-YOLO-M** and **YOLOv8-l** share the best Mask_AP. At the more challenging IoU = 0.5:0.95 setting, **YOLOv9-e** delivers the top Box_AP and jointly shares the best Mask_AP with **Hyper-YOLO-l** and **Hyper-YOLO-x**.

Comparison between End-to-End Models and Single-Stage Real-Time Models Comparing Table 5 with Table 6, the advantages of single-stage real-time models become substantially more evident. Under comparable model scales, single-stage architectures require fewer parameters and FLOPs, particularly when contrasted with complicated end-to-end models such as **ConvNeXtV2**, **Swin-Transformer**, and **Cswin-Transformer**. In terms of instance segmentation accuracy, their superiority is even more pronounced. Across all four mAP metrics, the second-best single-stage model already surpasses the top-performing end-to-end model. The gap becomes striking under stricter IoU conditions: on Box_AP_{0.5:0.95} and Mask_AP_{0.5:0.95}, the leading single-stage model (**YOLOv9-e**) exceeds the best end-to-end models (**ConvNeXtV2-Large** and **EMO-6M**) by 3.2% and 2.5%, respectively. This performance margin is primarily attributed to the markedly better handling of the more challenging *crack* category in both localization (Box-related metrics) and segmentation (Mask-related metrics).

To further illustrate these trends, Fig. 11 visualizes the strongest model from each series using a six-metric radar chart. The solid hexagons (single-stage models) consistently envelop the dashed hexagons (end-to-end models), clearly indicating the uniformly stronger localization and segmentation performance achieved by single-stage models. These differences should be understood as architecture-dependent benchmark observations under the unified CUBIT-InSeg protocol, rather than performance gains caused by the dataset itself.

5.3.2. Network attributes affecting instance segmentation

The computational efficiency of an algorithm is another key factor for practical deployment. Tables 5 and 6 summarize the model size (parameters or Params.), computational complexity (FLOPs), and inference speed (FPS) of all evaluated architectures, as well. Params. denotes the number of trainable parameters in a neural network, which also reflects the storage footprint of the trained model. FLOPs measure the computational cost required for a single forward pass, indicating how

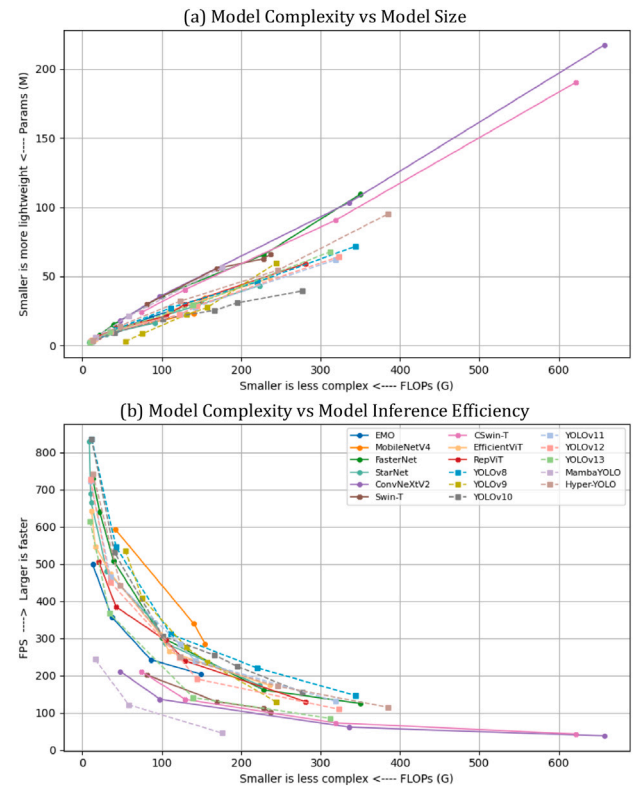


Fig. 12. (a) Model Complexity (FLOPs) vs Model Size (Params) and (b) Model Complexity (FLOPs) vs Inference Efficiency (FPS) on the test set of CUBIT-InSeg dataset. Solid line correspond to end-to-end models, while dashed line represent single-stage real-time models.

many floating-point operations the model must perform. FPS (frames per second) provides the most direct assessment of runtime inference efficiency.

In general, models with fewer parameters tend to require fewer FLOPs and are therefore expected to achieve higher FPS, as shown in Fig. 12. However, real-world inference performance is jointly influenced by several additional factors: (1) the extent to which the network architecture can fully utilize GPU Tensor Cores and CUDA kernels; (2) the level of hardware-specific optimization achieved by the deployment backend; (3) the computational overhead of post-processing steps such as NMS in Algorithm 1, which may increase when smaller inputs produce more small-object predictions. As a result, parameters, FLOPs, and FPS should be interpreted together to provide a more reliable and comprehensive assessment of the practical efficiency of each model family.

End-to-End Models **StarNet** series benefits from its lightweight design philosophy that relies on fewer layers and highly efficient element-wise multiplication. Consequently, **StarNet-s50** and **StarNet-s100** stand out as the two most compact models, each containing fewer than 3M parameters with only 8.9G and 10.4G FLOPs, respectively. As expected, **StarNet-s50** also delivers the highest inference speed, reaching 828.82 FPS on our Ubuntu 22.04 workstation. It is also noteworthy that **FasterNet-t0**, despite not being the smallest model, attains the second-highest throughput (729.19 FPS), owing to its partial convolution mechanism that significantly improves operator efficiency.

Single-Stage Real-Time Models Among all YOLO-family variants, the smallest architectures are **YOLOv13-n** and **YOLOv12-n**, with parameters of 2.70M and 2.81M, and FLOPs of 10.0G and 10.2G, respectively. Interestingly, the fastest models are not these smallest variants but rather **YOLOv10-n** and **YOLOv8-n**, achieving 836.54

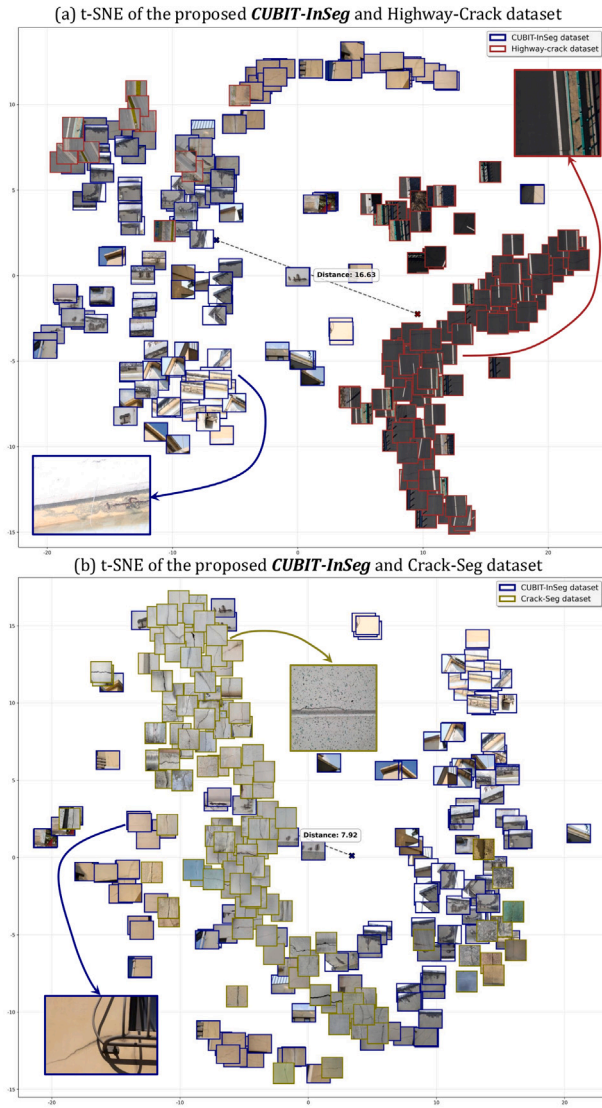


Fig. 13. t-SNE visualization of the proposed CUBIT-InSeg and (a) Highway-Crack [14] and (b) Crack-Seg [15] datasets.

FPS and 835.53 FPS. In contrast, although Mamba-YOLO-T has relatively small parameters and FLOPs, its inference speed is considerably lower due to the 2D-Selective-Scan [70], which introduces sequential dependencies.

Comparison between End-to-End and Single-Stage Real-Time Models Under the same model scaling factors, single-stage real-time architectures consistently exhibit lower parameters and FLOPs while achieving substantially higher FPS, which can be more intuitively seen in Fig. 12. This efficiency gap is particularly evident when comparing with complex end-to-end models such as ConvNeXtV2, Swin-Transformer, CSwin-Transformer, highlighting the strong suitability of single-stage models for real-time defect inspection applications.

5.3.3. Zero-shot validation on cross-domain datasets

One practical way to further examine the usefulness of a new dataset is to evaluate whether models trained on it can transfer to other related datasets without additional fine-tuning. Therefore, after completing the benchmark evaluation of over 80 individual networks on the proposed CUBIT-InSeg dataset, we further assess the cross-domain generalization ability of our models under a zero-shot setting. Specifically, we directly

apply the trained models to the test set parts of two publicly available unmanned systems collected datasets, Highway-Crack [14] for pure highway data and Crack-Seg [15] for building and pavement mixed data, without any training and fine-tuning.

To quantitatively and visually analyze the domain shift between datasets, Fig. 13 presents a t-SNE computed from color-histogram descriptors. For each image, 32-bin RGB histograms are extracted, concatenated, and normalized to form a compact appearance descriptor. These histogram features are standardized and optionally compressed using Principal component analysis (PCA) for improved stability, and then projected into a two-dimensional embedding space via t-SNE. The resulting distributions reveal clear structural differences between CUBIT-InSeg and the external datasets. To further quantify the discrepancy, we compute the Euclidean distance between the cluster centers of the two t-SNE embeddings as a simple domain-gap metric. The distances are 16.63 for Highway-Crack and 7.92 for Crack-Seg. These discrepancies in feature-space geometry highlight the inherent challenges of achieving robust defect instance segmentation under diverse real-world conditions, while also suggesting that models trained on CUBIT-InSeg exhibit promising transferability to related infrastructure inspection scenarios under the zero-shot setting.

Zero-Shot on Highway-Crack Dataset Following the default evaluation protocol of Highway-Crack [14], all input images are resized to 512×512 during zero-shot testing. Table 8 reports the results of end-to-end models. Notably, the transformer-based EfficientViT-M5 achieves the strongest zero-shot generalization, ranking first across all four metrics. For box-level evaluation, EMO-5M and FasterNet-1 rank second under Box_AP_{0.5} and Box_AP_{0.5:0.95}, respectively, whereas FasterNet-m and ConvNeXtV2-Large obtain the second-best performance on Mask_AP_{0.5} and Mask_AP_{0.5:0.95}.

Table 9 summarizes the zero-shot performance of single-stage real-time models. In addition to the consistently strong YOLOv9-e, the YOLOv8 family shows competitive cross-domain robustness: YOLOv8-1 ranks second on Box_AP_{0.5}, while YOLOv8-x ranks second on Box_AP_{0.5:0.95} and Mask_AP_{0.5}.

Overall, combining the results from Tables 8 and 9, we observe that Highway-Crack exhibits typical UAV-view highway scenes with pronounced illumination variations and small, distant crack targets. Under this challenging cross-domain setting, larger-capacity variants (m, b, l, x) generally achieve more stable zero-shot performance across both end-to-end and single-stage architectures, while the performance differences among the strongest models of each model families remain relatively moderate.

Zero-Shot on Crack-Seg Dataset For the Crack-Seg dataset, we follow its default setting and resize all input images to 416×416 during zero-shot testing. Table 10 summarizes the results of the end-to-end models. CSwin-Transformer-Large and EfficientViT-M5 achieve the highest Box_AP_{0.5} and Box_AP_{0.5:0.95}, respectively. Interestingly, despite being the smallest variant, CSwin-Transformer-Tiny obtains the best performance on two mask-based metrics, suggesting that compact transformer architectures can remain competitive under certain cross-domain conditions.

Table 11 shows the results for single-stage real-time models. Under IoU = 0.5, Mamba-YOLO-L and YOLOv10-x deliver the highest Box_AP and Mask_AP, respectively. Under the more demanding IoU (from 0.5 to 0.95), YOLOv13-x, despite its relatively modest in-domain performance, outperforms all competitors, achieving the best Box_AP and Mask_AP, with the latter being 0.22% higher than the second-best model YOLOv9-s.

Taken together, Tables 10 and 11 indicate that, unlike Highway-Crack, the Crack-Seg dataset does not strictly follow the trend where larger models generalize better. Several lightweight models (n, s), such as CSwin-Transformer-Tiny, YOLOv9-s, YOLOv10-s, and Hyper-YOLO-s, achieve top-two performance on multiple metrics, revealing a different behavior for this cross-domain ground vehicle-captured data.

Table 8

Zero-shot evaluation of selected end-to-end models on the test set of Highway-Crack dataset [14].

Models	Input Size	Box_AP ^{test} _{0.5} ↑	Box_AP ^{test} _{0.5:0.95} ↑	Mask_AP ^{test} _{0.5} ↑	Mask_AP ^{test} _{0.5:0.95} ↑
Conv-based Models					
EMO-1M (n) [57]	512 × 512	80.9%	54.8%	64.8%	21.2%
EMO-2M (s) [57]	512 × 512	81.7%	56.8%	68.9%	23.5%
EMO-5M (m) [57]	512 × 512	83.1% ₂	58.5%	72.5%	25.4%
EMO-6M (l) [57]	512 × 512	82.9%	59.7%	73.6%	26.6%
MobileNetV4Conv-S (s) [56]	512 × 512	78.0%	54.9%	63.8%	22.6%
MobileNetV4Conv-M (m) [56]	512 × 512	78.1%	55.5%	65.7%	22.8%
MobileNetV4Conv-L (l) [56]	512 × 512	79.3%	56.6%	68.2%	23.7%
FasterNet-t0 (n) [55]	512 × 512	77.4%	55.2%	64.3%	22.4%
FasterNet-t1 (n) [55]	512 × 512	78.9%	56.4%	68.5%	24.2%
FasterNet-t2 (n) [55]	512 × 512	79.8%	56.8%	69.8%	24.9%
FasterNet-s (s) [55]	512 × 512	81.4%	58.7%	72.2%	27.1%
FasterNet-m (m) [55]	512 × 512	82.6%	59.5%	74.0% ₂	26.8%
FasterNet-l (l) [55]	512 × 512	82.7%	59.8% ₂	72.5%	26.9%
Starnet-s50 (n) [54]	512 × 512	77.0%	53.6%	60.4%	19.7%
Starnet-s100 (n) [54]	512 × 512	78.1%	55.2%	64.9%	22.7%
Starnet-s150 (n) [54]	512 × 512	77.1%	54.9%	65.8%	22.3%
Starnet-s1 (s) [54]	512 × 512	79.2%	56.8%	69.4%	23.7%
Starnet-s2 (m) [54]	512 × 512	81.2%	58.3%	71.4%	26.5%
Starnet-s3 (l) [54]	512 × 512	80.7%	57.7%	71.7%	26.3%
Starnet-s4 (x) [54]	512 × 512	82.2%	59.2%	71.0%	26.1%
ConvNeXtV2-nano (n)	512 × 512	78.7%	56.5%	69.4%	26.3%
ConvNeXtV2-tiny (s)	512 × 512	78.8%	56.9%	69.3%	25.5%
ConvNeXtV2-base (m)	512 × 512	78.1%	57.1%	70.0%	27.1%
ConvNeXtV2-large (l)	512 × 512	79.7%	58.3%	73.5%	27.9% ₂
Transformer-based Models					
Swin-Transformer-Tiny (n) [59]	512 × 512	74.8%	53.5%	62.8%	21.9%
Swin-Transformer-Small (s) [59]	512 × 512	76.9%	55.0%	66.0%	24.3%
Swin-Transformer-Base (m) [59]	512 × 512	78.5%	56.6%	69.9%	26.5%
Swin-Transformer-Large (l) [59]	512 × 512	79.1%	57.3%	69.3%	26.9%
CSwin-Transformer-Tiny (n) [60]	512 × 512	78.7%	56.5%	69.4%	24.7%
CSwin-Transformer-Small (s) [60]	512 × 512	81.3%	58.1%	71.4%	26.5%
CSwin-Transformer-Base (m) [60]	512 × 512	81.8%	59.1%	73.1%	27.7%
CSwin-Transformer-Large (l) [60]	512 × 512	82.2%	59.7%	73.7%	27.8%
EfficientViT-M0 (n) [62]	512 × 512	78.2%	55.5%	66.3%	23.1%
EfficientViT-M1 (n) [62]	512 × 512	79.1%	56.6%	68.2%	23.7%
EfficientViT-M2 (s) [62]	512 × 512	80.6%	58.1%	68.5%	24.4%
EfficientViT-M3 (m) [62]	512 × 512	81.8%	59.0%	72.4%	26.6%
EfficientViT-M4 (l) [62]	512 × 512	81.8%	59.3%	73.8%	27.8%
EfficientViT-M5 (x) [62]	512 × 512	83.3% ₁	60.7% ₁	74.7% ₁	28.2% ₁
RepViT-m09 (n) [61]	512 × 512	79.3%	56.6%	69.7%	23.9%
RepViT-m10 (s) [61]	512 × 512	80.8%	58.4%	71.1%	25.7%
RepViT-m11 (m) [61]	512 × 512	80.9%	58.5%	71.7%	25.9%
RepViT-m15 (l) [61]	512 × 512	80.5%	58.7%	71.2%	26.6%
RepViT-m23 (x) [61]	512 × 512	81.5%	59.0%	71.7%	27.2%

(1) ↑ (↓) indicates that larger (smaller) values lead to better (worse) results. The best in orange, the second best are in blue.

Zero-Shot Performance Comparison between Two Cross-Domain Defect Datasets Fig. 13 shows that Crack-Seg is closer to CUBIT-InSeg in the t-SNE space, indicating a smaller distribution gap under the adopted histogram-based descriptors. This observation is broadly consistent with the zero-shot instance segmentation results of the selected models, where Crack-Seg yields slightly better overall performance than Highway-Crack. Specifically, under the easier IoU = 0.5 criterion, the selected models perform marginally better on Highway-Crack in both Box_AP and Mask_AP, as evidenced in Fig. 16(a) and (c), where the solid line with ▽ markers lies slightly above the dashed line with △ markers. In contrast, under the stricter IoU = 0.5:0.95 criterion, the Box_AP and Mask_AP on Highway-Crack decrease more noticeably and become lower than those on Crack-Seg. This trend is shown in Fig. 16(b) and (d), where the dashed line with △ markers is generally above the solid line with ▽ markers, with a visible margin.

This behavior may be related to several dataset-level characteristics. Beyond the coarse appearance gap reflected by the t-SNE embeddings in Figs. 13, 17 further compares the target-size and target-location statistics of the two external datasets. Fig. 17(a) and (c) show that crack instances in both datasets are predominantly elongated, while Crack-Seg contains relatively more targets whose width and height are closer to each other. Fig. 17(b) and (d) show that Crack-Seg targets are concentrated more strongly around the image center, whereas Highway-Crack targets are distributed more evenly across the image plane. Together with the smaller t-SNE distance between Crack-Seg and our CUBIT-InSeg, these dataset-level differences likely help explain why zero-shot transfer on Crack-Seg is slightly more favorable, especially under the stricter IoU = 0.5:0.95 criterion. They also indicate that the reported zero-shot evaluation is non-trivial, since both external datasets

differ from CUBIT-InSeg not only in overall appearance statistics but also in instance-level spatial and scale-related characteristics.

Figs. 14 and 15 present qualitative instance segmentation results on the two cross-domain test sets. We select representative models that perform strongly on each dataset: **FasterNet**, **EfficientViT**, **YOLOv8**, and **YOLOv9** for Highway-Crack, as well as **ConvNeXtV2**, **CSwin-Transformer**, **Mamba-YOLO**, and **YOLOv13** for Crack-Seg. Overall, these visualizations suggest that models trained on CUBIT-InSeg can retain reasonable detection and segmentation capability on related unseen datasets without additional fine-tuning. The qualitative results therefore provide encouraging evidence of zero-shot transferability, although broader cross-domain validation would still be needed to support stronger generalization claims.

6. Real-world validation

To evaluate the effectiveness and practical applicability of the proposed dataset under real operational conditions, we conduct a field experiment that couples pixel-level façade defect segmentation with geospatial registration on an actual high-rise building. This section reports the experimental setup, the end-to-end processing pipeline from images to the DT, and the quantitative assessment of registration accuracy, from both physical and virtual perspectives.

6.1. Experimental setup

The experiment is conducted on a high-rise multi-storey logistics warehouse located in Sha Tin, Hong Kong. This building features

Table 9

Zero-shot evaluation of selected single-stage real-time models on the test set of Highway-Crack dataset [14].

Models	Input Size	Box_AP ^{est} _{0.5} ↑	Box_AP ^{est} _{0.5:0.95} ↑	Mask_AP ^{est} _{0.5} ↑	Mask_AP ^{est} _{0.5:0.95} ↑
YOLO Series Models					
YOLOv8(8.1)-n [38]	512 × 512	79.9%	55.0%	64.3%	21.8%
YOLOv8(8.1)-s [38]	512 × 512	81.3%	56.9%	67.4%	23.6%
YOLOv8(8.1)-m [38]	512 × 512	82.8%	59.0%	72.7%	26.6%
YOLOv8(8.1)-l [38]	512 × 512	84.7% ₂	60.8%	76.4%	28.5%
YOLOv8(8.1)-x [38]	512 × 512	84.3%	61.3% ₂	76.8% ₂	29.5%
YOLOv9-t [19]	512 × 512	81.0%	56.4%	66.0%	22.8%
YOLOv9-s [19]	512 × 512	83.3%	58.8%	71.4%	24.7%
YOLOv9-m [19]	512 × 512	84.3%	60.3%	75.0%	27.6%
YOLOv9-c [19]	512 × 512	84.4%	60.4%	75.2%	28.7%
YOLOv9-e [19]	512 × 512	85.3% ₁	62.0% ₁	77.2% ₁	30.4% ₁
YOLOv10-n [63]	512 × 512	80.9%	56.7%	66.1%	22.4%
YOLOv10-s [63]	512 × 512	83.6%	58.7%	71.5%	24.8%
YOLOv10-m [63]	512 × 512	83.3%	60.5%	75.3%	27.5%
YOLOv10-b [63]	512 × 512	84.4%	60.1%	74.5%	27.6%
YOLOv10-l [63]	512 × 512	84.3%	60.5%	74.7%	28.0%
YOLOv10-x [63]	512 × 512	84.5%	61.2%	75.2%	28.8%
YOLOv11(8.3)-n [64]	512 × 512	77.9%	55.0%	65.0%	21.9%
YOLOv11(8.3)-s [64]	512 × 512	80.8%	57.9%	70.0%	24.6%
YOLOv11(8.3)-m [64]	512 × 512	81.4%	58.3%	72.2%	26.4%
YOLOv11(8.3)-l [64]	512 × 512	82.8%	59.9%	73.7%	26.8%
YOLOv11(8.3)-x [64]	512 × 512	83.5%	60.5%	75.5%	28.6%
YOLOv12-n [65]	512 × 512	78.7%	56.1%	65.3%	22.4%
YOLOv12-s [65]	512 × 512	81.1%	58.4%	70.3%	24.5%
YOLOv12-m [65]	512 × 512	81.6%	59.0%	73.7%	27.1%
YOLOv12-l [65]	512 × 512	82.4%	59.6%	74.8%	27.5%
YOLOv12-x [65]	512 × 512	83.0%	59.9%	74.7%	27.6%
YOLOv13-n [66]	512 × 512	76.8%	54.6%	66.2%	23.3%
YOLOv13-s [66]	512 × 512	79.2%	57.3%	71.7%	26.7%
YOLOv13-l [66]	512 × 512	82.3%	59.7%	73.9%	28.6%
YOLOv13-x [66]	512 × 512	82.2%	60.3%	75.1%	29.2%
MambaYOLO-T [67]	512 × 512	79.0%	57.7%	69.2%	25.4%
MambaYOLO-B [67]	512 × 512	81.7%	59.7%	75.0%	28.2%
MambaYOLO-L [67]	512 × 512	82.1%	60.3%	74.6%	29.8% ₂
Hyper-YOLO-n [39]	512 × 512	78.3%	55.6%	65.1%	22.3%
Hyper-YOLO-s [39]	512 × 512	81.0%	58.6%	69.9%	25.0%
Hyper-YOLO-m [39]	512 × 512	82.8%	60.2%	73.7%	27.5%
Hyper-YOLO-l [39]	512 × 512	83.1%	61.0%	75.4%	29.3%
Hyper-YOLO-x [39]	512 × 512	83.6%	61.3% ₂	75.5%	29.5%

(1) ↑ (↓) indicates that larger (smaller) values lead to better (worse) results. The best in orange, the second best are in blue.

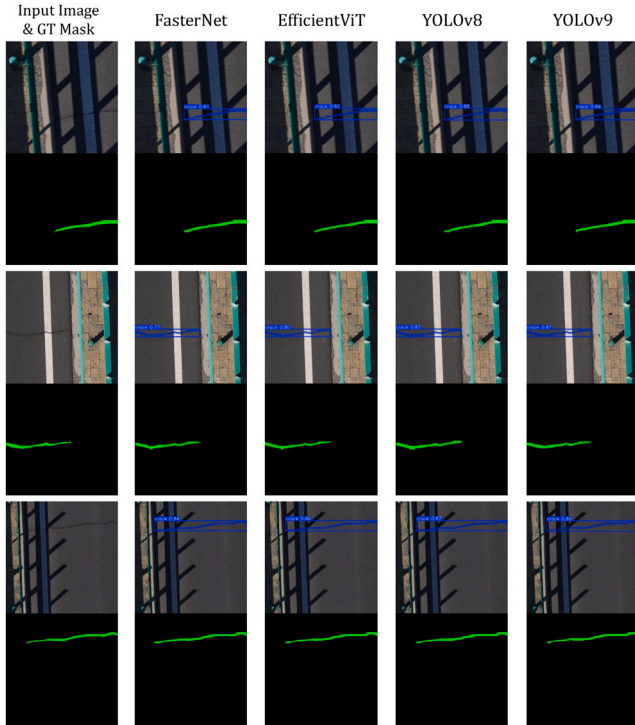


Fig. 14. Qualitative results of zero-shot instance segmentation based on models trained on the proposed CUBIT-InSeg dataset, evaluated on the Highway-Crack dataset [14].

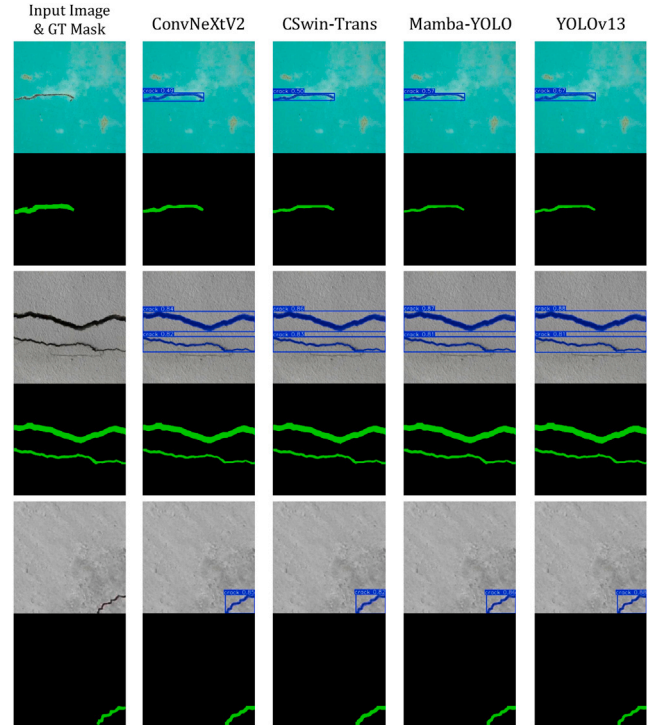


Fig. 15. Qualitative results of zero-shot instance segmentation based on models trained on the proposed CUBIT-InSeg dataset, evaluated on the Crack-Seg dataset [15].

Table 10

Zero-shot evaluation of selected end-to-end models on the test set of Crack-Seg dataset [15].

Models	Input Size	Box AP _{0.5} ^{test} ↑	Box AP _{0.5:0.95} ^{test} ↑	Mask AP _{0.5} ^{test} ↑	Mask AP _{0.5:0.95} ^{test} ↑
Conv-based Models					
EMO-1M (n) [57]	416 × 416	72.4%	60.1%	63.5%	28.2%
EMO-2M (s) [57]	416 × 416	76.5%	62.6%	61.9%	29.2%
EMO-5M (m) [57]	416 × 416	75.6%	64.3%	68.9%	29.8%
EMO-6M (l) [57]	416 × 416	74.9%	62.3%	66.7%	29.1%
MobileNetV4Conv-S (s) [56]	416 × 416	71.8%	61.6%	66.0%	28.3%
MobileNetV4Conv-M (m) [56]	416 × 416	72.6%	62.5%	63.9%	29.1%
MobileNetV4Conv-L (l) [56]	416 × 416	75.0%	62.7%	66.5%	30.5%
FasterNet-t0 (n) [55]	416 × 416	72.6%	63.3%	65.4%	27.8%
FasterNet-t1 (n) [55]	416 × 416	72.4%	62.2%	64.7%	27.4%
FasterNet-t2 (n) [55]	416 × 416	70.9%	60.8%	63.2%	27.9%
FasterNet-s (s) [55]	416 × 416	71.6%	62.5%	66.4%	30.2%
FasterNet-m (m) [55]	416 × 416	72.5%	62.1%	63.6%	30.1%
FasterNet-l (l) [55]	416 × 416	77.6% ₂	65.4%	69.8%	31.5%
Starnet-s50 (n) [54]	416 × 416	65.8%	53.7%	54.2%	18.5%
Starnet-s100 (n) [54]	416 × 416	72.8%	63.0%	64.9%	28.6%
Starnet-s150 (n) [54]	416 × 416	70.2%	61.2%	63.5%	28.1%
Starnet-s1 (s) [54]	416 × 416	71.9%	62.7%	67.0%	31.5%
Starnet-s2 (m) [54]	416 × 416	73.5%	63.1%	63.8%	30.2%
Starnet-s3 (l) [54]	416 × 416	77.4%	64.0%	66.8%	31.2%
Starnet-s4 (x) [54]	416 × 416	76.5%	65.2%	67.1%	31.7%
ConvNeXtV2-nano (n) [58]	416 × 416	73.9%	63.4%	68.2%	31.7%
ConvNeXtV2-tiny (s) [58]	416 × 416	73.0%	62.2%	68.3%	31.2%
ConvNeXtV2-base (m) [58]	416 × 416	75.6%	64.7%	66.7%	32.4%
ConvNeXtV2-large (l) [58]	416 × 416	75.8%	65.6% ₂	69.3%	32.8% ₂
Transformer-based Models					
Swin-Transformer-Tiny (n)	512 × 512	75.0%	63.7%	70.9% ₁	33.7% ₁
Swin-Transformer-Small (s)	512 × 512	76.8%	63.9%	68.9%	31.9%
Swin-Transformer-Base (m)	512 × 512	75.8%	64.1%	69.2%	31.9%
Swin-Transformer-Large (l)	512 × 512	76.7%	64.9%	65.7%	31.0%
CSwin-Transformer-Tiny (n) [60]	416 × 416	72.7%	63.1%	69.3%	30.6%
CSwin-Transformer-Small (s) [60]	416 × 416	74.8%	63.9%	69.9%	31.7%
CSwin-Transformer-Base (m) [60]	416 × 416	75.1%	63.6%	69.3%	32.4%
CSwin-Transformer-Large (l) [60]	416 × 416	77.9% ₁	65.6% ₂	70.1% ₂	32.2%
EfficientViT-M0 (n) [62]	416 × 416	72.3%	63.1%	64.7%	28.3%
EfficientViT-M1 (n) [62]	416 × 416	72.5%	63.2%	65.9%	29.2%
EfficientViT-M2 (s) [62]	416 × 416	71.3%	62.1%	66.1%	28.9%
EfficientViT-M3 (m) [62]	416 × 416	73.0%	63.0%	64.0%	31.0%
EfficientViT-M4 (l) [62]	416 × 416	74.1%	62.7%	66.6%	30.3%
EfficientViT-M5 (x) [62]	416 × 416	77.1%	66.0% ₁	66.5%	31.6%
RepViT-m09 (n) [61]	416 × 416	72.7%	62.5%	64.1%	28.7%
RepViT-m10 (s) [61]	416 × 416	71.8%	62.9%	67.2%	30.9%
RepViT-m11 (m) [61]	416 × 416	73.5%	63.3%	64.0%	30.8%
RepViT-m11 (l) [61]	416 × 416	76.7%	64.3%	68.4%	30.7%
RepViT-m23 (x) [61]	416 × 416	74.2%	63.9%	69.8%	30.8%

(1) ↑ (↓) indicates that larger (smaller) values lead to better (worse) results. The best in orange, the second best are in blue.

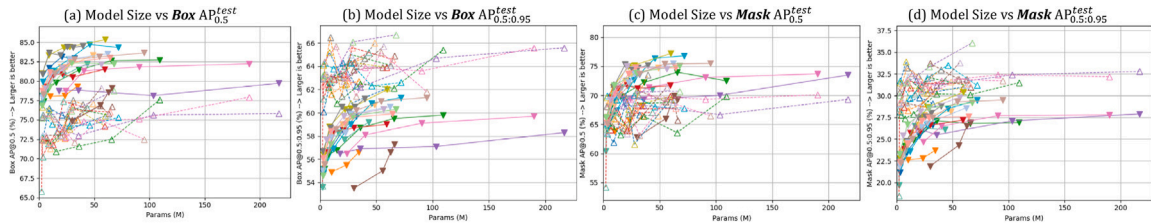


Fig. 16. Zero-shot instance segmentation results based on models trained on the proposed CUBIT-InSeg dataset, evaluated on the Highway-Crack and Crack-Seg test sets. Solid lines with inverted triangles ▽ denote Highway-Crack [14], while dashed lines with upright triangles △ denote Crack-Seg [15].

large reinforced-concrete façades with representative deterioration patterns, and therefore provides a suitable testbed for assessing automated façade inspection.

Image data are acquired using three *DJI Mavic 2 Pro* UAVs equipped with a high-resolution RGB camera (showed in Fig. 18). The multi-UAV data collection strategy, mentioned in Section 3.1, is utilized in this building to cover the four principal façades, namely the southwest (SW), southeast (SE), northeast (NE) and northwest (NW) walls, while maintaining consistent viewing geometry and sufficient image overlap. All images are stored with embedded EXIF metadata, including GPS position and altitude at the time of acquisition.

A three-dimensional DT of the building was reconstructed from the UAV imagery using a standard photogrammetric pipeline based on multi-view stereo (MVS) reconstruction [71]. The resulting georeferenced mesh provides the geometric reference frame for subsequent defect projection and quantitative error analysis. In the following, this

DT is treated as the ground-reference model onto which image-based detections are spatially anchored.

6.2. Defect segmentation and projection

The façade defects are inspected by YOLOv9-e [19], which is trained on our proposed CUBIT-InSeg dataset. For each UAV image, the model outputs a set of defect instances, where each instance consists of (1) a mask delineating the defect region at pixel resolution and (2) an associated axis-aligned bounding box. As illustrated in Fig. 19(a), these instances are overlaid on the original images as green contours and their labels.

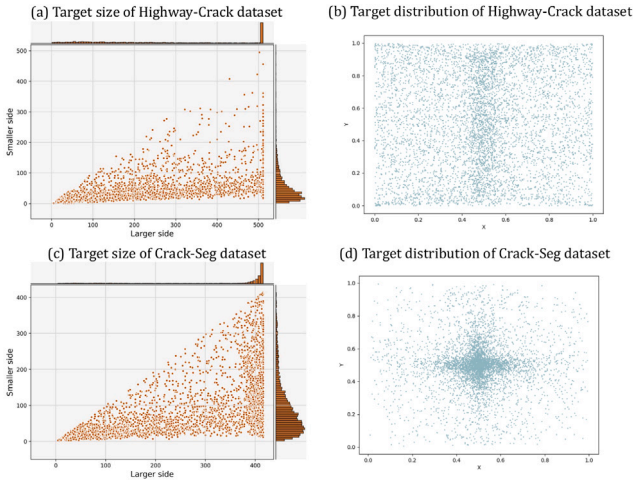
Although mask representation is advantageous for detailed appearance analysis, spatial registration to the DT is performed at the instance level. Specifically, for each detected defect, we compute the centroid of its bounding box in image coordinates and use this point as a compact

Table 11

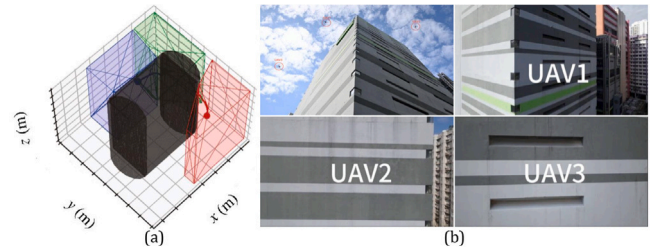
Zero-shot evaluation of the selected single-stage real-time models on the test set of Crack-Seg dataset [15].

Models	Input Size	Box_AP ^{est} _{0.5} ↑	Box_AP ^{est} _{0.5:0.95} ↑	Mask_AP ^{est} _{0.5} ↑	Mask_AP ^{est} _{0.5:0.95} ↑
YOLO Series Models					
YOLOv8(8.1)-n [38]	416 × 416	75.6%	64.2%	69.9%	31.7%
YOLOv8(8.1)-s [38]	416 × 416	74.4%	63.8%	69.2%	29.7%
YOLOv8(8.1)-m [38]	416 × 416	74.2%	64.3%	70.0%	32.0%
YOLOv8(8.1)-l [38]	416 × 416	74.3%	62.3%	68.0%	33.5%
YOLOv8(8.1)-x [38]	416 × 416	75.3%	62.6%	67.8%	31.2%
YOLOv9-t [19]	416 × 416	76.6%	64.1%	69.7%	31.5%
YOLOv9-s [19]	416 × 416	76.3%	65.2%	71.0%	33.9% ₂
YOLOv9-m [19]	416 × 416	73.9%	62.4%	67.6%	30.5%
YOLOv9-c [19]	416 × 416	73.8%	60.8%	61.6%	30.3%
YOLOv9-e [19]	416 × 416	75.5%	62.1%	69.3%	31.0%
YOLOv10-n [63]	416 × 416	75.3%	62.9%	68.1%	30.4%
YOLOv10-s [63]	416 × 416	75.6%	66.5% ₂	71.9% ₂	33.2%
YOLOv10-m [63]	416 × 416	73.0%	61.4%	69.9%	32.3%
YOLOv10-b [63]	416 × 416	75.5%	64.4%	67.0%	31.5%
YOLOv10-l [63]	416 × 416	77.5%	64.4%	70.1%	32.1%
YOLOv10-x [63]	416 × 416	77.0%	64.7%	73.5% ₁	33.0%
YOLOv11(8.3)-n [64]	416 × 416	73.0%	63.0%	63.8%	29.1%
YOLOv11(8.3)-s [64]	416 × 416	71.9%	63.1%	67.2%	32.1%
YOLOv11(8.3)-m [64]	416 × 416	73.6%	63.5%	64.2%	30.6%
YOLOv11(8.3)-l [65]	416 × 416	76.6%	65.0%	68.6%	32.1%
YOLOv11(8.3)-x [65]	416 × 416	75.6%	64.1%	68.8%	31.2%
YOLOv12-n [65]	416 × 416	72.5%	62.4%	65.4%	29.2%
YOLOv12-s [65]	416 × 416	72.1%	63.1%	67.8%	31.7%
YOLOv12-m [65]	416 × 416	73.9%	63.0%	64.2%	31.4%
YOLOv12-l [65]	416 × 416	76.3%	65.6%	67.6%	31.4%
YOLOv12-x [65]	416 × 416	76.0%	64.6%	67.8%	31.4%
YOLOv13-n [66]	416 × 416	75.2%	63.2%	64.2%	30.0%
YOLOv13-s [66]	416 × 416	75.6%	62.8%	67.1%	31.5%
YOLOv13-l [66]	416 × 416	76.1%	66.1%	70.6%	33.3%
YOLOv13-x [66]	416 × 416	78.7% ₂	66.7% ₁	70.7%	36.1% ₁
MambaYOLO-T [67]	416 × 416	73.0%	63.3%	65.1%	29.2%
MambaYOLO-B [67]	416 × 416	73.0%	62.7%	63.4%	28.8%
MambaYOLO-L [67]	416 × 416	79.1% ₁	65.7%	71.4%	33.8%
Hyper-YOLO-n [39]	416 × 416	75.6%	64.4%	68.0%	31.1%
Hyper-YOLO-s [39]	416 × 416	75.1%	65.7%	71.9% ₂	32.6%
Hyper-YOLO-m [39]	416 × 416	72.1%	61.4%	63.1%	30.3%
Hyper-YOLO-l [39]	416 × 416	75.0%	65.2%	69.1%	31.9%
Hyper-YOLO-x [39]	416 × 416	72.5%	61.8%	66.5%	31.7%

(1) ↑ (↓) indicates that larger (smaller) values lead to better (worse) results. The best in orange, the second best are in blue.

**Fig. 17.** Target size and target distribution of Highway-Crack [14] and Crack-Seg [15] datasets.

geometric proxy for the instance. This design is consistent with the methodology described in Section 4.1, where the image formation process is modeled by the calibrated pinhole camera, and back-projection of image points is carried out via the known intrinsic and estimated extrinsic parameters. In this paper, no explicit offline intrinsic calibration or per-image distortion correction was performed. Instead, metric conversion was based on the hardware parameters of the *DJI Mavic 2 Pro* camera together with the approximately equal-distance imaging strategy. Given the typical stand-off distances in façade inspection and the predominance of observations near the central field of view, the

**Fig. 18.** Multi-UAV data collection for this target industrial logistics warehouse. (a) Three UAVs coverage path planning; (b) Different views from each UAV.

residual distortion was considered sufficiently limited for the present inspection setting. Moreover, images with obvious distortion, blur, or geometric inconsistency were removed during data cleaning.

Given the camera pose associated with each image (derived from EXIF GPS data and refined through the photogrammetric bundle adjustment), the defect centroid is back-projected as a 3D ray into the world coordinate system. The intersection of this ray with the corresponding façade surface of the DT is then computed, which yields the registered defect location in 3D space. This procedure effectively maps 2D pixel-level detections into 3D, while preserving their semantic labels and enabling subsequent GIS-based analysis.

6.3. Defect registration and visualization

Building on our previous work on AI-assisted geospatial defect mapping [13], the registered defect points are further organized and visualized within a GIS-compatible environment. For each defect instance, the 3D intersection point on the DT is stored together with its

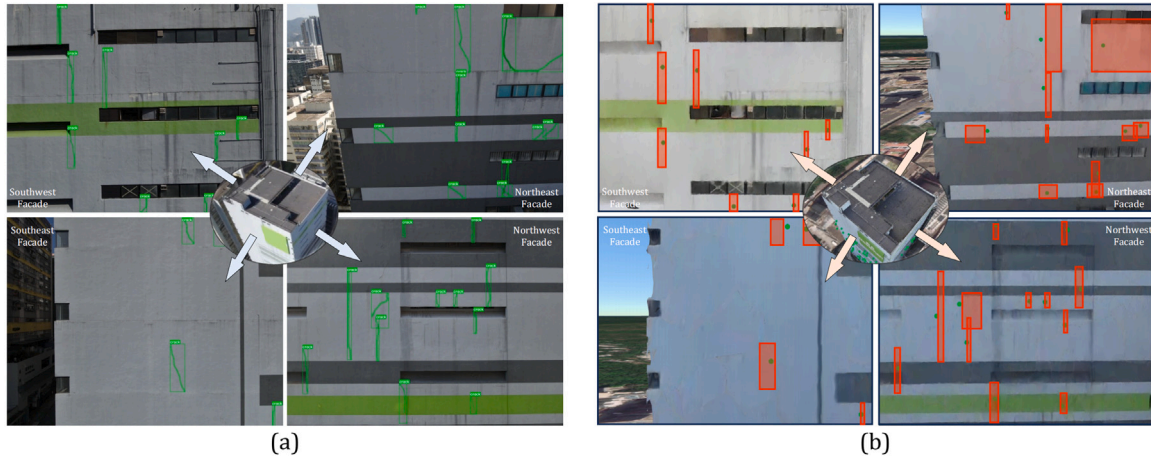


Fig. 19. Comparison between physical segmentation and virtual registration. (a) Raw aerial façade photos overlaid with pixel-level crack segmentation (green contours), while each quadrant shows one façade; (b) 3D building model with directional arrows pointing to each wall. Green dots are registered centers of defects; red rectangles are bounding boxes from segmentation results.

Table 12

Defect registration error for large-scale infrastructure.

Registration Error (cm)	Mean	MAE	RMSE	IQR
Horizontal	0.490	2.350	4.746	0
Vertical	0.592	1.037	2.385	0
Diagonal	1.360	4.056	7.149	3.747

semantic attributes (defect type, confidence score, and source image index). These points can then be queried, filtered, and aggregated using standard spatial operations.

Fig. 19 summarizes the overall effect of the proposed pipeline. Fig. 19(a) shows representative UAV images with pixel-level crack segmentation results (green masks and labels). Fig. 19(b) presents the reconstructed 3D building model, with arrows indicating the corresponding façades. It depicts the final registration outcome on the DT: red rectangles denote the projected bounding boxes of detected defects on the virtual façade, while green dots indicate the registered defect centroids. The close visual correspondence between physical images and their virtual counterparts demonstrates that the projection pipeline preserves both the spatial distribution and relative arrangement of defects across façades.

To quantitatively assess registration accuracy, we compute the horizontal, vertical, and diagonal (Euclidean) discrepancies between the registered defect points and their expected locations on the façade planes. The statistics reported in Table 12 include the mean error, mean absolute error (MAE), root-mean-square error (RMSE), and interquartile range (IQR). All values are expressed in centimetres. The low diagonal RMSE confirms that the proposed GIS-based registration is sufficiently accurate for subsequent defect aggregation and severity analysis at the façade-element level.

6.4. Physical-based SI quantification

After registering the detected defects onto the DT and visualizing their spatial distribution, we further evaluate the proposed framework from the perspective of physical-based severity assessment. The following analysis focuses on two aspects: defect-level physical measurement accuracy and SI-based severity assessment consistency, in order to examine whether pixel-level segmentation results can be reliably translated into engineering-relevant condition indicators.

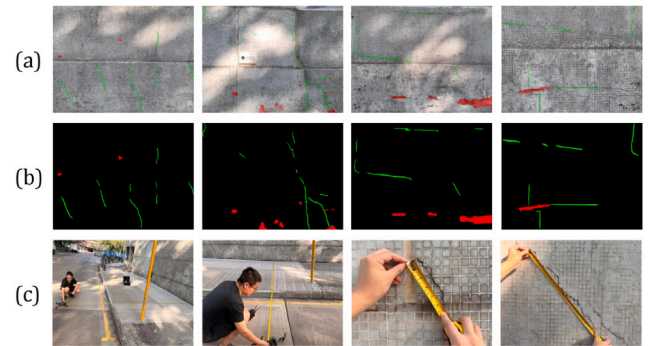


Fig. 20. Empirical validation of the proposed physical quantification pipeline: (a) Manually annotated ground-truth masks of cracks and spallings; (b) Model predicted instance masks of the corresponding defects; (c) On-site manual measurement of the corresponding defects.

6.4.1. Defect-level physical measurement

Since the subsequent SI assessment is built upon defect-level physical measurements, we first validate the reliability of the proposed pixel-to-physical quantification pipeline through an on-site measurement campaign at the China Resources Sheung Shui Warehouse in Hong Kong. A set of sampled crack and spalling defects (60 cracks and 14 spalling instances) were manually measured in the field and compared with the corresponding algorithmic predictions obtained from the instance masks. As shown in Fig. 20, the validation includes the ground-truth annotations, predicted masks, and field measurement setup. Quantitative comparison was performed using MAE, RMSE, and mean relative error for crack length, crack width, and spalling area, as summarized in Table 13. In addition, Fig. 21 presents scatter plots comparing the predicted and manually measured quantities with error bars, where most samples lie close to the identity line. Overall, these results indicate that the proposed quantification framework can produce physically meaningful measurements with sufficient accuracy for downstream SI-based condition assessment. Based on this validated quantification pipeline, we next evaluate whether the predicted defect masks can preserve the correct SI-based severity assessment relative to the ground truth.

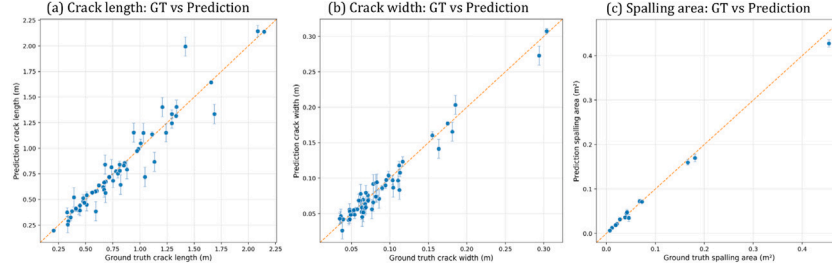


Fig. 21. Scatter plots comparing the predicted and manually measured defect quantities with error bars. Dashed line indicates the ideal agreement.

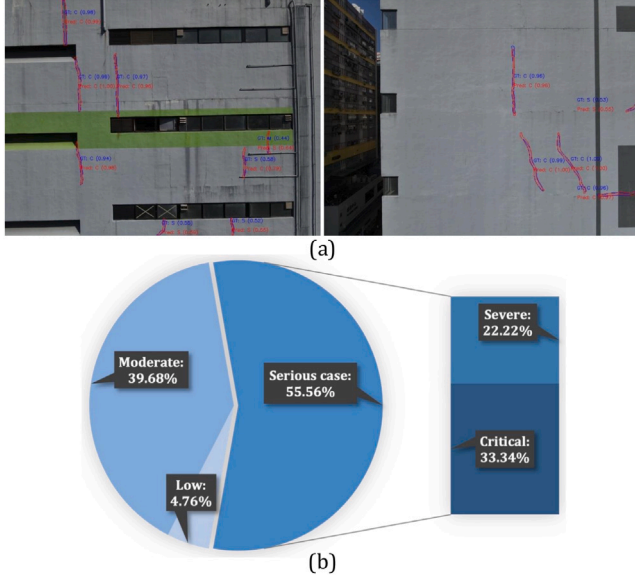


Fig. 22. (a) Quantified SI result of ground truth (blue contours) and predicted (red contours) labels; (b) Distribution of the defect SI level.

Table 13

Empirical validation results of physical quantification against manual measurements.

Defect type	Metric	MAE	RMSE	Mean relative error
Crack	Length (m)	0.07190	0.12500	8.85%
Crack	Width (m)	0.00846	0.01045	11.21%
Spalling	Area (m ²)	0.00577	0.00888	8.41%

6.4.2. SI assessment for individuals

After validating the physical measurement accuracy at the defect level, we further examine whether the predicted segmentation results can preserve the correct SI-based severity assessment relative to the ground truth. To this end, both the ground-truth annotations and the predicted defect masks are processed through the same physical quantification and SI computation pipeline. The resulting agreement is evaluated from two complementary perspectives: conventional pixel-level segmentation quality and engineering-oriented SI consistency.

The metrics we used are mIoU (visualized in Fig. 9) and mDice, whose equations are:

$$IoU_c = \frac{|GT_c \cap P_c|}{|GT_c \cup P_c|}, \quad mIoU = \frac{1}{C} \sum_{c=1}^C IoU_c; \quad (17)$$

$$Dice_c = \frac{2|GT_c \cap P_c|}{|GT_c| + |P_c|}, \quad mDice = \frac{1}{C} \sum_{c=1}^C Dice_c. \quad (18)$$

where c and C represent each defect class and the total number of defect classes, respectively.

Table 14

Defect instance segmentation results and assessment for large-scale infrastructure.

Metrics	mIoU	mDice	SI accuracy
Results	66.64%	79.95%	88.57%

Fig. 22(a) visualizes representative examples of SI quantification results overlaid on UAV images, where each defect instance is annotated with its corresponding severity level and SI value. Both ground truth and predicted defects are processed through the same physically based measurement and SI computation pipeline, ensuring that any difference in severity grading stems from segmentation discrepancies rather than post-processing bias.

Table 14 reports the corresponding quantitative results. In addition to conventional pixel-level segmentation measures (mIoU = 66.64% and mDice = 79.95%), we report SI accuracy, defined as the proportion of defect instances whose predicted severity level matches that derived from the ground truth. The SI accuracy of 88.57% indicates that, despite inevitable pixel-level deviations, most predicted defects are still assigned to the correct severity category. This result suggests that moderate segmentation errors do not necessarily translate into incorrect engineering-level condition assessment when physically grounded severity metrics are used.

The lower portion of Fig. 22(b) further illustrates the distribution of defect severity levels in the real-world dataset. Only 4.76% of the detected defects are classified as “Low”, while the remaining instances are distributed across “Moderate” (39.68%), “Severe” (22.22%), and “Critical” (33.34%) categories, while the severe and critical defects are regarded as serious cases. This skewed distribution reflects the challenging nature of the inspected façades, which predominantly exhibit non-trivial deterioration patterns rather than isolated hairline cracks. Such a distribution is consistent with field inspection reports for ageing high-rise buildings and underscores the necessity of prioritizing higher-severity defects in maintenance planning.

Importantly, the proposed SI-based evaluation provides a complementary perspective to traditional segmentation metrics. While IoU and Dice primarily measure geometric overlap at the pixel level, SI accuracy reflects the consistency of engineering-relevant decisions, namely defect severity grading and maintenance prioritization as shown in Fig. 23. This distinction is particularly important for real-world deployment, where inspection outcomes are ultimately used to support safety assessment, repair scheduling, and lifecycle management within a DT environment.

Overall, the real-world experiment demonstrates that the proposed framework can translate UAV-based instance-level defect segmentations into physically meaningful and DT-integrated severity assessments with good robustness. The strong agreement between ground-truth-based and prediction-based SI results further confirms the practical suitability of the proposed physically based quantification pipeline for façade inspection and decision support applications.

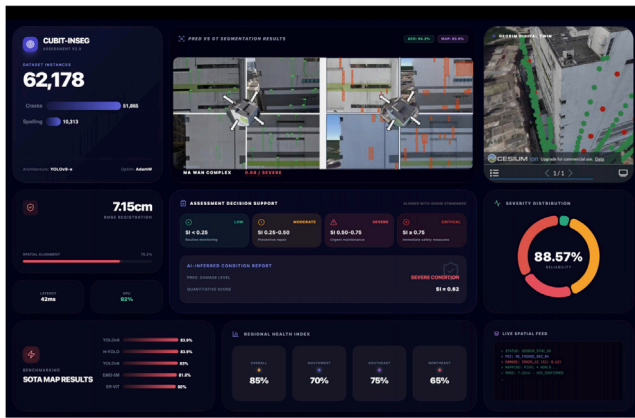


Fig. 23. Defect management platform for SI assessment.

7. Conclusions

In summary, recent advances in vision-based façade inspection have demonstrated the potential of deep learning for defect recognition, yet practical deployment remains constrained by limited dataset representativeness, weak cross-domain generalization, and insufficient linkage between pixel-level predictions and building-scale asset representations. The results presented in this paper indicate that these gaps can be jointly addressed by combining transferable instance-level perception with spatially grounded DT integration. Within this context, our contribution includes both an instance-level façade defect dataset and a validated pathway for translating recognition results into geometrically consistent, building-scale inspection outcomes.

First, we introduce CUBIT-InSeg, a multi-class instance segmentation dataset for building façade defects, mainly captured by UAV platforms in realistic urban environments and annotated at the pixel-instance level following established façade defect standards and professional inspection guidelines. To validate the usability and benchmark value of CUBIT-InSeg, we conduct large-scale training-from-scratch experiments on more than 80 models, together with zero-shot inference on two public defect datasets without additional fine-tuning. These results provide a broad benchmark view of how different architectures perform on CUBIT-InSeg and suggest encouraging transferability of models trained on it. Building on this dataset and benchmark foundation, we further bridge pixel-level defect perception with geospatially consistent representation on a building-scale DT. Starting from high-resolution RGB imagery acquired by an off-the-shelf drone, the framework integrates instance-level defect segmentation, photogrammetry-based reconstruction, and GIS-oriented registration to project detected defects from image space onto the DT surface. Real-world experiments on a multi-storey logistics building in Hong Kong, together with verification by building engineers using small-sample façade annotations, show that the proposed system can generate geometrically consistent defect maps across multiple façades and achieve centimetre-level registration accuracy, which is adequate for infrastructure-scale condition assessment and maintenance planning.

Several limitations of the present work should also be acknowledged. First, the physical quantification workflow relies to some extent on an approximately equal-distance imaging assumption. This assumption is partially supported by the explore-then-exploit path planning strategy, in which an initial surface model is reconstructed and then used to generate viewpoints approximately along local outward normals. As a result, the framework is not restricted to strictly planar façades and can also accommodate moderately curved or irregular surfaces when local geometry can be reconstructed reliably. Preliminary observations on a curved-surface case also suggest that the workflow

remains applicable when local geometry is sufficiently regular. However, its performance may degrade in scenes with highly complex geometries, such as dense steel-frame structures, severe self-occlusion, or strongly fragmented surface features, where both surface modeling and stable equal-distance imaging become more difficult. Second, the current dataset and experiments focus only on cracks and spalling. Although these defect types are highly relevant and representative in façade inspection practice, other common façade defects, such as corrosion stains, efflorescence, delamination, and water leakage, are not yet included. Third, the data are collected within a specific regional inspection context and primarily on concrete façades, meaning that the current distribution is inevitably shaped by local building typologies, material systems, climatic exposure conditions, and inspection practice. Accordingly, the generalizability of the dataset and the associated workflow to other geographical regions and façade material systems still requires further validation.

Future work will therefore focus on extending the dataset and workflow along several directions. A larger volume of UAV imagery will be collected across more diverse façade geometries, material types, defect categories, and environmental conditions to strengthen both dataset coverage and model robustness in real-world deployment. In parallel, we will explore quantification strategies that are less dependent on approximately equal-distance imaging and local planar assumptions, so as to improve applicability in more geometrically complex inspection scenarios. Beyond spatial diversity, enriching the temporal dimension of the data is also important. Long-term and repeated observations of the same buildings would enable the analysis of defect progression over time, thereby supporting a deeper understanding of deterioration mechanisms and providing stronger evidence for maintenance prioritization and intervention planning. Nevertheless, the present work establishes a solid foundation for instance-level façade defect recognition, engineering-oriented quantification, and DT-integrated building inspection, with clear value for both future research and practical asset management.

CRedit authorship contribution statement

Benyun Zhao: Writing – original draft, Visualization, Methodology, Investigation, Formal analysis, Data curation, Conceptualization. **Jihan Zhang:** Writing – original draft, Visualization, Methodology, Investigation, Formal analysis, Conceptualization. **Yijun Huang:** Writing – original draft, Visualization, Methodology, Investigation, Formal analysis. **Xi Chen:** Writing – review & editing, Supervision, Project administration, Funding acquisition, Conceptualization. **Ben M. Chen:** Writing – review & editing, Resources, Project administration, Funding acquisition, Conceptualization.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Acknowledgments

This work was supported in part by the InnoHK of the Government of the Hong Kong Special Administrative Region via the Hong Kong Centre for Logistics Robotics (HKCLR), and in part by the Research Grants Council of Hong Kong SAR under Grants 14217922, 14209623, 14209424 and 14200524.

Data availability

The source data of CUBIT-InSeg dataset is open-sourced at <https://github.com/BenyunZhao/CUBIT-InSeg>.

References

- [1] Ali Akbar Shirzadi Javid, Parviz Ghoddousi, Gholamreza Ghodrati Amiri, Khalil Donyadideh, A photogrammetry method to study the relationship between thixotropy and bond strength of multi-layers casting of self-consolidating concrete, *Constr. Build. Mater.* 204 (2019) 530–540, <http://dx.doi.org/10.1016/j.conbuildmat.2019.01.204>.
- [2] Junzhi Zhang, Jin Huang, Chuanqing Fu, Le Huang, Hailong Ye, Characterization of steel reinforcement corrosion in concrete using 3d laser scanning techniques, *Constr. Build. Mater.* 270 (2021) 121402, <http://dx.doi.org/10.1016/j.conbuildmat.2020.121402>.
- [3] Wei Jiang, Youjun Xie, Jianxian Wu, Guangcheng Long, Influence of age on the detection of defects at the bonding interface in the crts iii slab ballastless track structure via the impact-echo method, *Constr. Build. Mater.* 265 (2020) 120787, <http://dx.doi.org/10.1016/j.conbuildmat.2020.120787>.
- [4] Haifeng Li, Nansha Li, Renbiao Wu, Huaichao Wang, Zhongcheng Gui, Dezhen Song, Gpr-rcnn: An algorithm of subsurface defect detection for airport runway based on gpr, *IEEE Robot. Autom. Lett.* 6 (2) (2021) 3001–3008, <http://dx.doi.org/10.1109/LRA.2021.3062599>.
- [5] Khashayar Asadi, Akshay Kalkunte Suresh, Alper Ender, Siddhesh Gotad, Suraj Maniyyar, Smit Anand, Mojtaba Noghabaei, Kevin Han, Edgar Lobaton, Tianfu Wu, An integrated ugv-uav system for construction site data collection, *Autom. Constr.* 112 (2020) 103068, <http://dx.doi.org/10.1016/j.autcon.2019.103068>.
- [6] Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xi-aohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, et al., An image is worth 16x16 words: Transformers for image recognition at scale, 2020, <http://dx.doi.org/10.48550/arXiv.2010.11929>, arXiv preprint.
- [7] Navneet Dalal, Bill Triggs, Histograms of oriented gradients for human detection, in: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2005, <http://dx.doi.org/10.1109/CVPR.2005.177>.
- [8] Jun Kang Chow, Kuan-fu Liu, Pin Siang Tan, Zhaoyu Su, Jimmy Wu, Zhaofeng Li, Yu-Hsing Wang, Automated defect inspection of concrete structures, *Autom. Constr.* 132 (2021) 103959, <http://dx.doi.org/10.1016/j.autcon.2021.103959>.
- [9] Qian Wang, Hongzhe Yue, Zeyu Li, Guanying Huang, Rui Liu, Jiaming Wang, Yolo-depth: leveraging depth information for improved instance segmentation in construction scenes, *Adv. Eng. Informatics* 69 (2026) 104048, <http://dx.doi.org/10.1016/j.aei.2025.104048>.
- [10] Tsung-Yi Lin, Michael Maire, Serge Belongie, James Hays, Pietro Perona, Deva Ramanan, Piotr Dollár, C. Lawrence Zitnick, Microsoft coco: Common objects in context, in: *Proceedings of the European Conference on Computer Vision*, Springer, 2014, pp. 740–755, http://dx.doi.org/10.1007/978-3-319-10602-1_48.
- [11] Guidong Yang, Kangcheng Liu, Jihan Zhang, Benyun Zhao, Zuoquan Zhao, Xi Chen, Ben M. Chen, Datasets and processing methods for boosting visual inspection of civil infrastructure: A comprehensive review and algorithm comparison for crack classification, segmentation, and detection, *Constr. Build. Mater.* 356 (2022) 129226, <http://dx.doi.org/10.1016/j.conbuildmat.2022.129226>.
- [12] Xinyi Wang, Chuanxiang Gao, Xi Chen, Ben M. Chen, Fast and secure distributed multi-agent coverage control with an application to infrastructure inspection and reconstruction, in: *Proceedings of the 42nd Chinese Control Conference*, 2023, pp. 5998–6005, <http://dx.doi.org/10.23919/CCS58697.2023.10241159>.
- [13] Jihan Zhang, Benyun Zhao, Guidong Yang, Xunkuai Zhou, Yijun Huang, Chuanxiang Gao, Xi Chen, Ben M. Chen, Ai-empowered digital twin modeling for high-precision building defect management integrating uav and geobim, *Build. Simul.* 18 (2025) 2531–2558, <http://dx.doi.org/10.1007/s12273-025-1332-9>.
- [14] Zhonghua Hong, Fan Yang, Haiyan Pan, Ruyan Zhou, Yun Zhang, Yanling Han, Jing Wang, Shuhu Yang, Peng Chen, Xiaohua Tong, et al., Highway crack segmentation from unmanned aerial vehicle images using deep learning, *IEEE Geosci. Remote. Sens. Lett.* 19 (2021) 1–5, <http://dx.doi.org/10.1109/LGRS.2021.3129607>.
- [15] Roboflow Universe Team, Crack dataset, 2022, <https://universe.roboflow.com/university-bswxt/crack-bphdr>.
- [16] British Standards Institution adoption of ISO standard, Buildings and constructed assets – service life planning – part 7: Performance evaluation for feedback of service life data from practice, 2017, <https://www.iso.org/standard/61149.html>.
- [17] Buildings Department, Guidelines on mandatory building inspection scheme, in: Volume 1: Pre-1980 Residential and Composite Buildings, 2017, <https://www.bd.gov.hk/doc/en/resources/codes-and-references/code-and-design-manuals/GGBISMWIS.pdf>.
- [18] Hong Kong Institute of Surveyors, Practice notes for building surveying and façade inspection, 2016, <https://www.hkis.org.hk/archive/materials/category/20160927174934.0.pdf>.
- [19] Chien-Yao Wang, I-Hau Yeh, Hong-Yuan Mark Liao, Yolov9: Learning what you want to learn using programmable gradient information, in: *Proceedings of the European Conference on Computer Vision*, Springer, 2024, pp. 1–21, http://dx.doi.org/10.1007/978-3-031-72751-1_1.
- [20] Markus Eisenbach, Ronny Stricker, Daniel Seichter, Karl Amende, Klaus Debes, Maximilian Sesselmann, Dirk Ebersbach, Ulrike Stoeckert, Horst-Michael Gross, How to get pavement distress detection ready for deep learning? a systematic approach, in: 2017 International Joint Conference on Neural Networks, IEEE, 2017, pp. 2039–2047, <http://dx.doi.org/10.1109/IJCNN.2017.7966101>.
- [21] Ronny Stricker, Markus Eisenbach, Maximilian Sesselmann, Klaus Debes, Horst-Michael Gross, Improving visual road condition assessment by extensive experiments on the extended gaps dataset, in: 2019 International Joint Conference on Neural Networks, IEEE, 2019, pp. 1–8, <http://dx.doi.org/10.1109/IJCNN.2019.8852257>.
- [22] Ronny Stricker, Dustin Aganian, Maximilian Sesselmann, Daniel Seichter, Marius Engelhardt, Roland Spielhofer, Matthias Hahn, Astrid Hautz, Klaus Debes, Horst-Michael Gross, Road surface segmentation-pixel-perfect distress and object detection for road assessment, in: 2021 IEEE 17th International Conference on Automation Science and Engineering, IEEE, 2021, pp. 1789–1796, <http://dx.doi.org/10.1109/CASE49439.2021.9551591>.
- [23] Fan Yang, Lei Zhang, Sijia Yu, Danil Prokhorov, Xue Mei, Haibin Ling, Feature pyramid and hierarchical boosting network for pavement crack detection, *IEEE Trans. Intell. Transp. Syst.* 21 (4) (2019) 1525–1535, <http://dx.doi.org/10.1109/TITS.2019.2910595>.
- [24] Qiwei Mei, Mustafa Gül, Md Riasat Azim, Densely connected deep neural network considering connectivity of pixels for automatic crack detection, *Autom. Constr.* 110 (2020) 103018, <http://dx.doi.org/10.1016/j.autcon.2019.103018>.
- [25] Hongyan Xu, Xiu Su, Yi Wang, Huaiyu Cai, Kerang Cui, Xiaodong Chen, Automatic bridge crack detection using a convolutional neural network, *Appl. Sci.* 9 (14) (2019) 2867, <http://dx.doi.org/10.3390/app9142867>.
- [26] Philipp Hühwohl, Ruodan Lu, Ioannis Brilakis, Multi-classifier for reinforced concrete bridge defects, *Autom. Constr.* 105 (2019) 102824, <http://dx.doi.org/10.1016/j.autcon.2019.04.019>.
- [27] Benyun Zhao, Xunkuai Zhou, Guidong Yang, Junjie Wen, Jihan Zhang, Jia Dou, Guang Li, Xi Chen, Ben M. Chen, High-resolution infrastructure defect detection dataset sourced by unmanned systems and validated with deep learning, *Autom. Constr.* 163 (2024) 105405, <http://dx.doi.org/10.1016/j.autcon.2024.105405>.
- [28] Christian Benz, Paul Debus, Huy Khanh Ha, Volker Rodehorst, Crack segmentation on uas-based imagery using transfer learning, in: 2019 International Conference on Image and Vision Computing New Zealand, 2019, pp. 1–6, <http://dx.doi.org/10.1109/IVCNZ48456.2019.8960998>.
- [29] C.F. Özgenel, Arzu Gönenç Sorguç, Performance comparison of pretrained convolutional neural networks on crack detection in buildings, in: *Proceedings of the 35th International Symposium on Automation and Robotics in Construction*, 2018, pp. 693–700, <http://dx.doi.org/10.22260/ISARC2018/0094>.
- [30] Hiroya Maeda, Yoshihide Sekimoto, Toshikazu Seto, Takehiro Kashiwayama, Hiroshi Omata, Road damage detection and classification using deep neural networks with smartphone images, *Computer-Aided Civ. Infrastruct. Eng.* 33 (12) (2018) 1127–1141, <http://dx.doi.org/10.1111/mice.12387>.
- [31] Hiroya Maeda, Takehiro Kashiwayama, Yoshihide Sekimoto, Toshikazu Seto, Hiroshi Omata, Generative adversarial network for road damage detection, *Computer-Aided Civ. Infrastruct. Eng.* 36 (1) (2021) 47–60, <http://dx.doi.org/10.1111/mice.12561>.
- [32] He Zhang, Leyuan Ma, Zhenmin Yuan, Hexu Liu, Enhanced concrete crack detection and proactive safety warning based on i-st-unet model, *Autom. Constr.* 166 (2024) 105612, <http://dx.doi.org/10.1016/j.autcon.2024.105612>.
- [33] Luyuan Wu, Yi Cheng, Aifang Su, Boyang Zhang, Zifa Wang, Jingbo Tong, Meng Li, Anqi Zhang, Multiscale semantic segmentation and damage assessment of village houses in post-flood scenarios using an enhanced deeplabv3 with dual attention mechanism, *Adv. Eng. Informatics* 68 (2025) 103759, <http://dx.doi.org/10.1016/j.aei.2025.103759>.
- [34] Shuqian Duan, Minghuan Zhang, Jiecheng Xiong, Shili Qiu, Zhengkun Xie, Quan Jiang, Yongyuan Kou, Dsyolo: A dynamic snake convolution-based segmentation model for tunnel lining cracks in complex environments, *Eng. Struct.* 344 (2025) 121325, <http://dx.doi.org/10.1016/j.engstruct.2025.121325>.
- [35] Kaiming He, Georgia Gkioxari, Piotr Dollár, Ross Girshick, Mask r-cnn, in: *Proceedings of the IEEE International Conference on Computer Vision*, 2017, pp. 2961–2969, <http://dx.doi.org/10.1109/ICCV.2017.322>.
- [36] Xizhou Zhu, Weijie Su, Lewei Lu, Bin Li, Xiaogang Wang, Jifeng Dai, Deformable detr: Deformable transformers for end-to-end object detection, 2020, <http://dx.doi.org/10.48550/arXiv.2010.04159>, arXiv preprint.
- [37] Bowen Cheng, Ishan Misra, Alexander G. Schwing, Alexander Kirillov, Rohit Girdhar, Masked-attention mask transformer for universal image segmentation, in: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2022, pp. 1290–1299, <http://dx.doi.org/10.1109/CVPR52688.2022.00135>.
- [38] Glenn Jocher, Jing Qiu, Ayush Chaurasia, Ultralytics yolov8, 2023, <https://github.com/ultralytics/ultralytics/tree/v8.1.9>.
- [39] Yifan Feng, Jiangang Huang, Shaoyi Du, Shihui Ying, Jun-Hai Yong, Yipeng Li, Guiguang Ding, Rongrong Ji, Yue Gao, Hyper-yolo: When visual object detection meets hypergraph computation, *IEEE Trans. Pattern Anal. Mach. Intell.* 47 (4) (2025) 2388–2401, <http://dx.doi.org/10.1109/TPAMI.2024.3524377>.
- [40] Alexander Kirillov, Eric Mintun, Nikhila Ravi, Hanzi Mao, Chloe Rolland, Laura Gustafson, Tete Xiao, Spencer Whitehead, Alexander C. Berg, Wan-Yen Lo, et al., Segment anything, in: *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 2023, pp. 4015–4026, <http://dx.doi.org/10.1109/ICCV51070.2023.00371>.
- [41] Zehao Ye, Lucy Lovell, Asaad Faramarzi, Jelena Ninić, Sam-based instance segmentation models for the automation of structural damage detection, *Adv. Eng. Informatics* 62 (2024) 102826, <http://dx.doi.org/10.1016/j.aei.2024.102826>.

- [42] Jiehui Wang, Tamon Ueda, Pujin Wang, Zhibin Li, Yong Li, Building damage inspection method using uav-based data acquisition and deep learning-based crack detection, *J. Civ. Struct. Health Monit.* 15 (1) (2025) 151–171, <http://dx.doi.org/10.1007/s13349-024-00836-3>.
- [43] Yang Lyu, Muqing Cao, Shenghai Yuan, Lihua Xie, Vision-based plane estimation and following for building inspection with autonomous uav, *IEEE Trans. Syst. Man, Cybern.: Syst.* 53 (12) (2023) 7475–7488, <http://dx.doi.org/10.1109/TSMC.2023.3299237>.
- [44] Jianxiao Zheng, Jixiang Tan, Gang Chen, Research on uav coverage path planning in building visual inspection, *J. Build. Eng.* (2025) 113556, <http://dx.doi.org/10.1016/j.jobe.2025.113556>.
- [45] Junjie Chen, Isabelle Chan, Ioannis Brilakis, Shifting research from defect detection to defect modeling in computer vision-based structural health monitoring, *Autom. Constr.* 164 (2024) 105481, <http://dx.doi.org/10.1016/j.autcon.2024.105481>.
- [46] Seongdeok Bang, Somin Park, Hongjo Kim, Hyoungkwan Kim, Encoder–decoder network for pixel-level road crack detection in black-box images, *Computer-Aided Civ. Infrastruct. Eng.* 34 (8) (2019) 713–727, <http://dx.doi.org/10.1111/mice.12440>.
- [47] Shang Jiang, Jian Zhang, Real-time crack assessment using deep neural networks with wall-climbing unmanned aerial system, *Computer-Aided Civ. Infrastruct. Eng.* 35 (6) (2020) 549–564, <http://dx.doi.org/10.1111/mice.12519>.
- [48] Gustavo H. Beckman, Dimos Polyzois, Young-Jin Cha, Deep learning-based automatic volumetric damage quantification using depth camera, *Autom. Constr.* 99 (2019) 114–124, <http://dx.doi.org/10.1016/j.autcon.2018.12.006>.
- [49] Krisada Chaiyasarn, Apichat Buatik, Hisham Mohamad, Mingliang Zhou, Sirisilp Kongsilp, Nakhorn Poovarodom, Integrated pixel-level cnn-fcn crack detection via photogrammetric 3d texture mapping of concrete structures, *Autom. Constr.* 140 (2022) 104388, <http://dx.doi.org/10.1016/j.autcon.2022.104388>.
- [50] Fardin Bahreini, Amin Hammad, Dynamic graph cnn based semantic segmentation of concrete defects and as-inspected modeling, *Autom. Constr.* 159 (2024) 105282, <http://dx.doi.org/10.1016/j.autcon.2024.105282>.
- [51] Benyun Zhao, Qigeng Duan, Guidong Yang, Jerry Tang, Zhenbo Song, Junjie Wen, Xuchen Liu, Qingxiang Li, Lei Lei, Jihan Zhang, et al., Lightweight yet high-performance defect detector for uav-based large-scale infrastructure real-time inspection, in: 2025 IEEE International Conference on Robotics and Automation, IEEE, 2025, pp. 13675–13682, <http://dx.doi.org/10.1109/ICRA55743.2025.11127758>.
- [52] Junjie Chen, Weisheng Lu, Jinfeng Lou, Automatic concrete defect detection and reconstruction by aligning aerial images onto semantic-rich building information model, *Computer-Aided Civ. Infrastruct. Eng.* 38 (8) (2023) 1079–1098, <http://dx.doi.org/10.1111/mice.12928>.
- [53] Kentaro Wada, Labelme: Image polygonal annotation with python, 2021, <http://dx.doi.org/10.5281/zenodo.5711226>.
- [54] Xu Ma, Xiyang Dai, Yue Bai, Yizhou Wang, Yun Fu, Rewrite the stars, in: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, 2024, pp. 5694–5703, <http://dx.doi.org/10.1109/CVPR52733.2024.00544>.
- [55] Jerun Chen, Shiu-Hong Kao, Hao He, Weipeng Zhuo, Song Wen, Chul-Ho Lee, S.-H. Gary Chan, Run, don't walk: Chasing higher flops for faster neural networks, in: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, 2023, pp. 12021–12031, <http://dx.doi.org/10.1109/CVPR52729.2023.01157>.
- [56] Danfeng Qin, Chas Lechner, Manolis Delakis, Marco Fornoni, Shixin Luo, Fan Yang, Weijun Wang, Colby Banbury, Chengxi Ye, Berkin Akin, et al., Mobilenetv4-universal models for the mobile ecosystem, 2024, http://dx.doi.org/10.1007/978-3-031-73661-2_5, arXiv preprint.
- [57] Jiangning Zhang, Xiangtai Li, Jian Li, Liang Liu, Zhucun Xue, Boshen Zhang, Zhengkai Jiang, Tianxin Huang, Yabiao Wang, Chengjie Wang, Rethinking mobile block for efficient attention-based models, in: 2023 IEEE/CVF International Conference on Computer Vision, IEEE Computer Society, 2023, pp. 1389–1400, <http://dx.doi.org/10.1109/ICCV51070.2023.00134>.
- [58] Sanghyun Woo, Shoubhik Debnath, Ronghang Hu, Xinlei Chen, Zhuang Liu, In So Kweon, Saining Xie, Convnext v2: Co-designing and scaling convnets with masked autoencoders, in: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, 2023, pp. 16133–16142, <http://dx.doi.org/10.1109/CVPR52729.2023.01548>.
- [59] Ze Liu, Yutong Lin, Yue Cao, Han Hu, Yixuan Wei, Zheng Zhang, Stephen Lin, Baining Guo, Swin transformer: Hierarchical vision transformer using shifted windows, in: Proceedings of the IEEE/CVF International Conference on Computer Vision, 2021, pp. 10012–10022, <http://dx.doi.org/10.1109/ICCV48922.2021.00986>.
- [60] Xiaoyi Dong, Jianmin Bao, Dongdong Chen, Weiming Zhang, Nenghai Yu, Lu Yuan, Dong Chen, Baining Guo, Cswin transformer: A general vision transformer backbone with cross-shaped windows, in: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, 2022, pp. 12124–12134, <http://dx.doi.org/10.48550/arXiv.2107.00652>.
- [61] Ao Wang, Hui Chen, Zijia Lin, Hengjun Pu, Guiguang Ding, Repvit: Revisiting mobile cnn from vit perspective, in: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, 2024, pp. 15909–15921, <http://dx.doi.org/10.1109/CVPR52733.2024.01506>.
- [62] Xinyu Liu, Houwen Peng, Ningxin Zheng, Yuqing Yang, Han Hu, Yixuan Yuan, Efficientvit: Memory efficient vision transformer with cascaded group attention, in: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, 2023, pp. 14420–14430, <http://dx.doi.org/10.48550/arXiv.2305.07027>.
- [63] Ao Wang, Hui Chen, Lihao Liu, Kai Chen, Zijia Lin, Jungong Han, Guiguang Ding, YOLOv10: Real-time end-to-end object detection, 2024, <http://dx.doi.org/10.48550/arXiv.2405.14458>, arXiv preprint.
- [64] Glenn Jocher, Jing Qiu, Ultralytics yolov11, 2024, <https://github.com/ultralytics/ultralytics/tree/v8.3.10>.
- [65] Yunjie Tian, Qixiang Ye, David Doermann, YOLOv12: Attention-centric real-time object detectors, 2025, <http://dx.doi.org/10.48550/arXiv.2502.12524>, arXiv preprint.
- [66] Mengqi Lei, Siqi Li, Yihong Wu, Han Hu, You Zhou, Xinhui Zheng, Guiguang Ding, Shaoyi Du, Zongze Wu, Yue Gao, YOLOv13: Real-time object detection with hypergraph-enhanced adaptive visual perception, 2025, arXiv preprint, <https://arxiv.org/abs/2506.17733>.
- [67] Zeyu Wang, Chen Li, Huiying Xu, Xinzhou Zhu, Hongbo Li, Mamba yolo: A simple baseline for object detection with state space model, in: Proceedings of the AAAI Conference on Artificial Intelligence, vol. 39, 2025, pp. 8205–8213, <http://dx.doi.org/10.1609/aaai.v39i8.32885>.
- [68] Jan Hosang, Rodrigo Benenson, Bernt Schiele, Learning non-maximum suppression, in: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2017, <http://dx.doi.org/10.1109/CVPR.2017.685>.
- [69] Kai Chen, Jiaqi Wang, Jiangmiao Pang, Yuhang Cao, Yu Xiong, Xiaoxiao Li, Shuyang Sun, Wansen Feng, Ziwei Liu, Jiarui Xu, et al., Mmdetection: Open mmlab detection toolbox and benchmark, 2019, <http://dx.doi.org/10.48550/arXiv.1906.07155>, arXiv 2019. arXiv preprint.
- [70] Yue Liu, Yunjie Tian, Yuzhong Zhao, Hongtian Yu, Lingxi Xie, Yaowei Wang, Qixiang Ye, Jianbin Jiao, Yunfan Liu, Vmamba: Visual state space model, in: Advances in Neural Information Processing Systems, vol. 37, 2024, pp. 103031–103063, <http://dx.doi.org/10.52202/079017-3273>.
- [71] Guidong Yang, Benyun Zhao, Jihan Zhang, Junjie Wen, Qingxiang Li, Lei Lei, Xi Chen, Ben M. Chen, Det-recon-reg: An intelligent framework toward automated uav-based large-scale infrastructure inspection, *IEEE Trans. Instrum. Meas.* 74 (2025) 1–16, <http://dx.doi.org/10.1109/TIM.2025.3571118>.